PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

Ref-2

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :
H04K 1/04

A1

(11) International Publication Number: WO 93/03559

(43) International Publication Date: 18 February 1993 (18.02.93)

(21) International Application Number: PCT/US92/06497

(22) International Filing Date: 4 August 1992 (04.08.92)

(30) Priority data:
741,097 5 August 1991 (05.08.91) US

(71) Applicant: TELEDYNE INDUSTRIES, INC. [US/US]; 19601 Nordhoff Street, Northridge, CA 91324 (US).

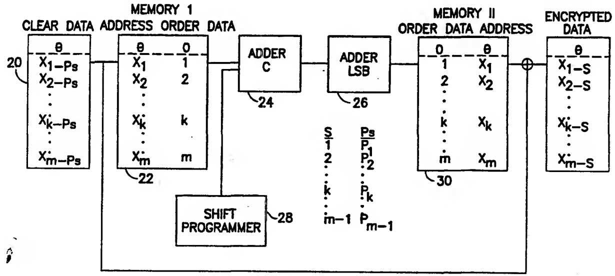
(72) Inventor: MITTENTHAL, Lothrop ; 240 Los Padres Drive, Thousand Oaks, CA 91361 (US).

(74) Agents: BLAKELY, Roger, W., Jr. et al.; Blakely, Sokoloff, Taylor and Zafman, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 90025 (US).

(81) Designated States: AT, AU, BB, BG, BR, CA, CH, CS, DE, DK, ES, FI, GB, HU, JP, KP, KR, LK, LU, MG, MN, MW, NL, NO, PL, RO, RU, SD, SE, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, SN, TD, TG).

Published
With international search report.

(54) Title: NONLINEAR DYNAMIC SUBSTITUTION DEVICES AND METHODS FOR BLOCK SUBSTITUTIONS



(57) Abstract

Method and apparatus for nonlinearizing modulo 2 addition (24) based encryption by block substitution techniques which allows use of the substitution scheme with relatively simple hardware and yet makes cryptanalysis more difficult. The basic block substitution (22), a one to one mapping of n bit binary numbers onto themselves, is based on the fact that certain permutations of the n bit binary numbers define a block substitution by modulo 2 addition (24) of one permuted set of numbers to another, and that a subset of these define equations having an additive relationship when viewed as vectors. This allows the simple changing of the transformation on a frequent basis. Then the equations are nonlinearized, also in an orderly (34) and readily variable manner, so that the remainder of the set equations may no longer be generated from a limited subset of the equations. Various properties of the transformations and methods of using the same are disclosed.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FI	Finland	MN	Mongolia
AU	Australia	FR	France	MR	Mauritania
BB	Barbados	GA	Gabon	MW	Malawi
BE	Belgium	GB	United Kingdom	NL	Netherlands
BF	Burkina Faso	GN	Guinea	NO	Norway
BG	Bulgaria	GR	Greece	NZ	New Zealand
BJ	Benin	HU	Hungary	PL	Poland
BR	Brazil	IE	Ireland	PT	Portugal
CA	Canada	IT	Italy	RO	Romania
CF	Central African Republic	JP	Japan	RU	Russian Federation
CG	Congo	KP	Democratic People's Republic of Korea	SD	Sudan
CH	Switzerland			SE	Sweden
CI	Côte d'Ivoire	KR	Republic of Korea	SK	Slovak Republic
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CS	Czechoslovakia	LK	Sri Lanka	SU	Soviet Union
CZ	Czech Republic	LU	Luxembourg	TD	Chad
DE	Germany	MC	Monaco	TG	Togo
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America

NONLINEAR DYNAMIC SUBSTITUTION DEVICES AND METHODS FOR BLOCK SUBSTITUTIONS

RELATED APPLICATION

This application is a continuation-in-part of application Serial No. 07/416,953 filed October 4, 1989.

BACKGROUND OF THE INVENTION

1. Field of the Invention.

The present invention relates to the field of encryption devices and methods, and more particularly, to block substitution encryption methods and devices.

2. Prior Art.

In many cases, it is desired to communicate information in digital form from one location to another in a manner which is clear and unambiguous to the receiver, but which is incomprehensible to an interloper therebetween. Accordingly, in many instances, it is common to encrypt the information to be communicated by some predetermined encryption process, to transmit the encrypted form of the information and to then decrypt the information at the receiving location. Depending upon the degree of security desired, a relatively simple and easily broken encryption may be used, as any level of encryption will make the transmission meaningless to the casual interloper. In other situations, the degree of security desired may dictate the use of an encryption technique which is more difficult to decipher by cryptanalysis, or of course hopefully in the highest level of security, make the same substantially impossible to decipher. Applications for such encryption techniques include commercial applications such as sensitive communications between manufacturing plants, bank branches, etc., and military applications including but not limited to IFF (identification friend or foe). While in some cases the primary objective of the encryption is to prevent an interloper from deciphering the information being communicated, in other cases a primary object, such as in IFF, is to prevent the interloper from himself originating false information with the same encryption scheme so as to mislead the intended receiver. Both objectives are frequently present in many applications.

Block substitution is a method used to encrypt a clear text message which is in the form of a sequence of binary numbers. In accordance with the method, the sequence is broken into blocks of some predetermined block length n, with the block substitution device substituting a unique new block of binary numbers for each of those in the clear text. Substitute blocks constitute the encrypted message or cipher text, each substitute block representing a nonambiguous one-to-one transformation of a clear text block. In the prior art, such substitution generally takes place by means of look-up tables, switching arrangements, or feedback shift registers. However, without changing codes or substitution schemes frequently, the encryption may be broken by cryptanalysis, though, changing look-up tables is cumbersome, only a limited number of possible switching arrangements is practical, and repeated cycling of a shift register is time consuming. An additional problem arises in finding substitutions which do not have any pattern or bias in them. At the present time, candidate substitutions are examined by computer simulation for possible systematic patterns and in some cases, additional circuitry is used to compensate therefor.

Various types of encryption equipment and methods are well-known in the prior art. See for instance U.S. Patents No. 3,796,830, 3,798,359, 4,078,152, 4,195,200, 4,255,811, 4,316,055 and 4,520,232. In general, these systems as they relate to block substitution are key dependent ciphering and deciphering systems and are not based upon block substitution by modulo 2 addition of one additive permuted set of numbers to another, as in the present invention.

In the parent application, methods and apparatus for modulo 2 addition based encryption by block substitution techniques were disclosed which allow use of the substitution scheme with relatively simple hardware. The block substitution, a one to one mapping of n bit binary numbers onto themselves, is based on the fact that certain permutations of the n bit binary numbers define a block substitution by modulo 2 addition of one permuted set of numbers to another, and that a subset of these define equations having an additive relationship when viewed as vectors, whereby the remainder of the set may be generated from a limited subset of the equations. This allows the simple changing of the transformation on a frequent basis. Various properties of the transformations and methods of using the same were disclosed. The fact that the remainder of the set equations may be generated from a limited subset of the equations, however, may make cryptanalysis less difficult than desired in some applications. Accordingly the present invention comprises a method and apparatus for nonlinearizing the equations, also in an orderly and readily variable manner, so that the remainder of the set equations may no longer be generated from a limited subset of the equations.

BRIEF DESCRIPTION OF THE INVENTION

Methods and apparatus for nonlinearizing modulo 2 addition based encryption by block substitution techniques which allows use of the substitution scheme with relatively simple hardware and yet makes cryptanalysis more difficult. The basic block substitution, a one to one mapping of n bit binary numbers onto themselves, is based on the fact that certain permutations of the n bit binary numbers define a block substitution by modulo 2 addition of one permuted set of numbers to another, and that a subset of these define equations having an additive relationship when viewed as vectors. This allows the simple changing of the transformation on a frequent basis. Then the equations are nonlinearized, also in an orderly and readily variable manner, so that the remainder of the set equations may no longer be generated from a limited subset of the equations. Various properties of the transformations and methods of using the same are disclosed.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a many-one transformation of one set of three bit binary numbers to another set of binary numbers by a modulo 2 addition.

Figure 2 illustrates a one-to-one transformation of one set of three bit binary numbers to another set of binary numbers by a modulo 2 addition.

Figure 3 presents the transformation equations of Figure 2 reordered, excluding the first equation, making the three digit number in the first column the same as the three digit number in the second column of the preceding row. Excluding the first equation, each column now is in the same order but with different starting positions.

Figure 4 corresponds to Figure 3, though with the first and third columns shifted vertically with respect to the second column. These shifts are 6 and 2 positions downward respectively. Except for the first equation, each column remains in the same order but with different starting positions.

Figure 5 corresponds to Figure 4, with the $\oplus$ and = symbols interchanged for encryption purposes.

Figure 6 is a block diagram of an apparatus for encrypting data.

Figure 7 is a block diagram of apparatus for decrypting data encrypted by the apparatus of Figure 6.

Figure 8 is an example of encryption using Figure 6.

Figure 9 is an example of decryption using Figure 7.

Figure 10 presents a set of transformation equations corresponding to those of Figure 4 with the fixed word 001 added to columns 1 and 2 thereof. Except for the first equation, columns 1 and 2 are in the same order but with different starting positions.

Figure 11 is a block diagram for an apparatus for encrypting data in accordance with a set of transformation equations such as those of Figure 10.

Figure 12 is a block diagram for an apparatus for decrypting data encrypted with the apparatus of Figure 11.

Figure 13 is an example of encryption using Figure 11.

Figure 14 is an example of encryption using Figure 12.

Figure 15 illustrates a rearrangement equations in Figure 2 into the general form $x_{n-1} \oplus x_n = z_n$.

Figure 16 is a block diagram of a system for encryption and decryption in accordance with the present invention.

Figure 17 illustrates a set of equations useful for encryption and decryption derived by adding the offset 0101 to the first and second columns of the set of equations presented on page A12 of Appendix 2.

DETAILED DESCRIPTION OF THE INVENTION

Since the present invention comprises methods and apparatus for nonlinearizing the modulo 2 addition based encryption by block substitution described in detail in parent application, that disclosure is repeated herein to provide a basis for the improvement of the present invention. Thus as described in the parent application, in the description to follow, the methods and apparatus of the parent application will first be described with respect to blocks of n bit binary numbers where the value of n is 3. Then the methods and apparatus will be expanded to n bit blocks generally, and certain characteristics of blocks up to n = 8 will be presented. By presenting the following example for n = 3, it is believed that the concepts of the original invention may be better understood than would be the case if a larger block having many more combinations were used.

Block substitution is the term usually applied to a one-to-one mapping of the n-bit binary numbers onto themselves. This mapping can be written as a pairing of the 2^n n-bit numbers:

$$\begin{array}{cc}
X_1 & Z_1 \\
X_2 & Z_2 \\
\cdot & \cdot \\
\cdot & \cdot \\
X_k & Z_k \\
\cdot & \cdot \\
\cdot & \cdot \\
\cdot & \cdot
\end{array}$$

where each column is the set of the same 2^n distinct n-bit numbers but written in different orders. Thus, this mapping can be thought of as a permutation of the n-bit numbers written as:

$$\begin{array}{ccc} X_1 & X_2 & ..X_k \\ Z_1 & Z_2 & ..Z_k \end{array}$$

or $(X_1\ X_i\ X_j)$ for some set of indices. This usual notation for permutations simply means that $X_1 \longrightarrow X_i$, $X_i \longrightarrow X_j$, etc.

Going back to the column notations, one could define a set of simple equations from the original set and its image:

$$\begin{array}{ccccc} Y_1 & \oplus & X_1 & = & Z_1 \\ Y_2 & \oplus & X_2 & = & Z_2 \\ & \cdot & & & \cdot \\ & \cdot & & & \cdot \\ & \cdot & & & \cdot \\ Y_k & \oplus & X_k & = & Z_k \\ & \cdot & & & \cdot \\ & \cdot & & & \cdot \\ & \cdot & & & \cdot \end{array}$$

where $\oplus$ means modulo 2 addition (i.e., addition of corresponding digits without any carry). In general, the set $\{Y_1, Y_2, ...\}$ will not all be distinct, but in certain circumstances they will be. In accordance with the original invention, when they are distinct, block substitutions can be generated by modulo 2 addition rather than by conventional means. The main tasks are to determine the circumstances, if any, in which this scheme works, how the substitutions can be quickly changed, and the lack of bias.

It is not obvious that block substitutions can ever be generated by modulo 2 addition. For example, consider the attempt to substitute one arrangement of 3-bit binary numbers for another by modulo 2 addition shown in Figure 1. In column 3 on the right, 011 and 100 each appear twice, while 001 and 110 never appear. The numbers in column 1 on the left, acting on the numbers in column 2 in the center, constitute a transformation of the set of 3-bit binary words of column 1 into themselves. This is a many-one transformation and is useless for block substitutions because of the ambiguity that results when trying to recover the original block for the transformed blocks 011 and 100.

Trying another arrangement as shown in Figure 2 gives a different result. Any pair of columns now constitutes a one-to-one transformation. In particular, the transformation is one-to-one from the 3-bit binary numbers of column 3 (the clear text) onto themselves, the encrypted text of column 1. Each column consists of all the 3-bit numbers exactly once.

Obviously, one could use the transformations of Figure 2 to transform any three digit binary block into an encrypted binary block, and of course use the same equations to de-encrypt the encrypted message by finding the encrypted word in column 1 and then selecting the corresponding clear text word in the same row, column 3 of Figure 2. This is most convenient if ⊕ and = are interchanged as shown in Figure 5. An equivalent transformation to transform the encrypted word back to the clear text word results if the words of column one are added to those of column two to obtain those in column three.

Referring again to Figure 2, an interesting property of the transformation shown therein, and for that matter, for all transformations of the type of interest herein, may be seen. In particular, of the eight blocks of three binary numbers, the lower four blocks 000, 001, 010 and 011 map into two blocks of the lower four, namely 000 and 001, and two blocks of the upper four, namely 110 and 111. Similarly of course, the four larger blocks of the eight map two into blocks in the lower four, and two into blocks of the upper four. Similarly, the even blocks 000, 010, 100 and 110 map into two even blocks, 000 and 010, and into two odd blocks, 001 and 011. The odd four blocks map half into odd blocks and half into even blocks. Obviously for decryption, the same is true. Thus, knowledge of some characteristic of the encrypted block such as its being large, small, even, odd, etc., does not convey any similar knowledge of a characteristic of the unencrypted block. As a result of this, the encryption is said to be unbiased. For that matter it should be noted that, even considering the middle digit of each block, the four blocks of Figure 2 having a zero as the middle digit map two blocks into blocks also having a zero as a middle digit, and two blocks having one as the middle digit. Similarly of course, the four blocks having a one as the middle digit map two into blocks having a one as a middle digit, and two into blocks having a zero as the middle digit. This property applies to all block sizes and extends to dividing equally all set of blocks which may be characterized algebraically as maximal subgroups. This unbiased character of the encryption is a highly beneficial characteristic of the encryption scheme disclosed herein, particularly in conjunction with the frequent changing of the encryption from time to time.

In particular, in any practical encryption device of course, one would like to be able to frequently change the encryption scheme so that patterns will not persist for a sufficient length of time to allow cryptanalysis of the pattern. For this purpose, certain properties of the equations of Figure 2 may be recognized by rearranging the rows of Figure 2 as shown in Figure 3. Rearrangement of the rows in any manner of course does not effect the transformation in any way, as each of the equations maintains its own integrity separate and apart from its position in the table. In essence, the second row of Figure 3 is the 4th row in Figure 2 and the third row of Figure 3 is the fifth row in Figure 2, with each successive row being arranged so that the left column in each successive row contains the same 3 bit number as the second column of the preceding row. When so arranged, neglecting the first or identity row, it will be noted that each of the three columns contains the same sequence of the 3 bit binary numbers, with wrap-around. In particular, the first column has the same sequence as the second column, but displaced downward therefrom one position (or upward six positions), and the third column has the same sequence as the second column, though displaced downward three positions (or upward four positions) from the sequence of column two.

Neglecting the first row or identity row of Figure 3 again, if the 3 bit binary numbers in column 1 are shifted downward a total of six positions with wrap-around with respect to the second column, it will be noted that a one to one transformation still results, as shown in Figure 4. Except for the identity row, the transformation is entirely different from that of Figure 3. By way of example, 111 column 3 maps into 011 column 1 in Figure 3, and maps into 100 column 1 in Figure 4. In addition however, it is important to note that the sequence of the 3 digit numbers in columns 1 and 3 of Figure 4 (separating out the identify row) is still the same as that in column 2 of Figures 3 and 4, though each is shifted with wrap-around in comparison to column 2. Thus, the transformation of Figure 3 has been changed to the new transformation of Figure 4 by merely shifting the numbers in the first column of Figure 3 with respect to those in the second column, and with the numbers in the third column also being shifted with respect to those in the second column, but by a different amount to preserve the integrity of the modulo 2 addition equations. Again, for decryption, symbols ⊕ and = of Figure 4 can be interchanged as in Figure 5.

More generally for any block size, the sets of equations can be written as:

ENCRYPTION

$$\begin{array}{ccccc}
\underline{1} & \underline{2} & & & \underline{3} \\
\theta & = & \theta & \oplus & \theta \\
X_{1-s} & = & X_1 & \oplus & X_{1-Ps} \\
X_{2-s} & = & X_2 & \oplus & X_{2-Ps} \\
\vdots & & \vdots & & \vdots \\
X_{k-s} & = & X_k & \oplus & X_{k-Ps} \\
\vdots & & \vdots & & \vdots \\
X_{m-s} & = & X_m & \oplus & X_{m-Ps}
\end{array}$$

$\theta = 00...00$

DECRYPTION

$$\begin{array}{ccccc}
\underline{1} & \underline{2} & & & \underline{3} \\
\theta & \oplus & \theta & = & \theta \\
X_{1-s} & \oplus & X_1 & = & X_{1-Ps} \\
X_{2-s} & \oplus & X_2 & = & X_{2-Ps} \\
\vdots & & \vdots & & \vdots \\
X_{k-s} & \oplus & X_k & = & X_{k-Ps} \\
\vdots & & \vdots & & \vdots \\
X_{m-s} & \oplus & X_m & = & X_{m-Ps}
\end{array}$$

For block size n, $m = 2^n-1$. $\theta = 00...00$, the n bit word consisting of all zeroes.

If column 1 is shifted by S positions with respect to column 2, then column 3 is shifted by a different amount P_s to preserve the integrity of the modulo 2 addition equations. For a given shift S, P_s is determined by the shift programmer.

Now referring to figure 6, a block diagram of a system for carrying out encryption in accordance with the encryption and decryption techniques discussed so far may be seen.

The clear test word is sent to its address in Memory I. This corresponds to selecting a word X_{k-Ps} from column 3 other than θ. The concept is to add it to its counterpart in column 2. If X_{k-Ps} is other than θ and is to be added to X_k, this is equivalent to adding the word with order data K - P_s in column 3 to the word with order K - P_s + P_s = K, also in column 3. Thus the order data of the clear test word K - P_s is sent to the adder to be added to P_s. The new order number is sent to its address in Memory II. The content of that address is added modulo 2 to the clear test word to obtain the encrypted word
X_{k-s} in column 1. If the clear text word is θ, its cipher test image is the same.

Adding of the order data is accomplished by two adders, carry (C) and least significant bit (LSB). The carry adder adds the numbers conventionally with carry, e.g. 001 + 011 = 100. However, if the addition requires more than n digits, that is, a 1 is carried to the n+1 position, that extra 1 is instead added to the first position, e.g., 100 + 110 = 1010 => 011. This is accomplished by the LBS adder. This is simply addition modulo m where $m = 2^n - 1$. In this example, $n = 3$, $m = 7$ and the addition expressed in decimal terms is $4 + 6 = 10 \equiv 3 \bmod 7$ where 100 => 4, 110 => 6 and 011 => 3.

The block diagram for decryption is shown at Figure 7. The cipher text word is sent to its address in Memory I. This corresponds to selecting a word X_{k-s} from column 1 other than θ. The concept is to add it to its counterpart, X_k in column 2. This is equivalent to adding X_{k-s} in column 1 to the word with order data $K - s + s = K$, also in column 1. Thus the order data of the cipher text word, K-s is sent to the adder to be added to s. The new order number is sent to its address in Memory II. The contents of that address is added modulo 2 to the cipher text word to obtain the de-encrypted word X_{k-Ps} in column 3. If the cipher text word is θ, it is de-encrypted as θ.

The addition of order data, $K - S + S$ and $K - P_s + P_s$ is understood to be modulo m or with wraparound. That is, if the order data is greater than m, the last position, m is subtracted from the order data. If the cipher text word is θ, it is de-encrypted as the same word.

The shift program determines the order in which the shifts, S, in column 1 are used, with the corresponding P_s shift S in column 3. Any desired order can be used. The shift S corresponds to a power of the basic permutation described on Page 8, which determines the substitution by addition.

Thus, by way of example, in Figure 8, if the clear data value is 010, then that address in Memory I provides order data 001, which is binary notation that 010 is in position 1 in the sequence in Memory I (column 3 of the set of equations). The first shift position in the program is S = 6, for which $P_6 = 2$. To the position of 010, $K - P_6 = 1$ is added $P_6 = 2$. In binary notation, 001 + 010 = 011. Corresponding to the address 011 in Memory II is the number 100. (This is equivalent to saying that 100 is in position 3 in column 3). 110 = 100 ⊕ 010 is the cipher text word. This represents the first of the additive equations in Figure 5.

For decryption, the cipher text word is 110. In Figure 9, that address in Memory I provides order data 100, or position 4 in the sequence in Memory I. The first shift position in the program is S = 6. To the position of 110, K - 6 = 4, is added 6, or 110 in binary notation. 4 + 6 = 10. Subtracting by m = 7, 10 - 7 = 3, or position 3 with wrap around. In binary notation, 100 + 110 = 011 modulo 7. Corresponding to the address 011 in Memory II is the number 100. 110 ⊕ 100 = 010. This represents the first of the additive equations in Figure 4.

If one adds Modulo 2 a fixed number to the first and second columns of Figure 4. A still further one-to-one transformation results.

ENCRYPTION

1		2		3
$(I \oplus Y)$	$=$	$(I \oplus Y)$	$\oplus$	I
$(X_{1-s} \oplus Y)$	$=$	$(X_1 \oplus Y)$	$\oplus$	X_{1-Ps}
$(X_{2-s} \oplus Y)$	$=$	$(X_2 \oplus Y)$	$\oplus$	X_{2-Ps}
:		:		:
$(X_{k-s} \oplus Y)$	$=$	$(X_k \oplus Y)$	$\oplus$	X_{k-Ps}
:		:		:
$(X_{m-s} \oplus Y)$	$=$	$(X_m \oplus Y)$	$\oplus$	X_{m-Ps}

DECRYPTION

1		2		3
$(I \oplus Y)$	$\oplus$	$(I \oplus Y)$	$=$	I
$(X_{1-s} \oplus Y)$	$\oplus$	$(X_1 \oplus Y)$	$=$	X_{1-Ps}
$(X_{2-s} \oplus Y)$	$\oplus$	$(X_2 \oplus Y)$	$=$	X_{2-Ps}
:		:		:
$(X_{k-s} \oplus Y)$	$\oplus$	$(X_k \oplus Y)$	$=$	X_{k-Ps}
:		:		:
$(X_{m-s} \oplus Y)$	$\oplus$	$(X_m \oplus Y)$	$=$	X_{m-Ps}

Now referring to Figures 11 and 12, for any block size a block diagram for carrying out encryption and decryption using a fixed word other than θ, the zero word, may be seen. The procedure is essentially the same as before with the additional step of adding the fixed word Modulo 2 as the last step in the encryption process and the first step in the decryption process.

An example is shown in Figure 13 and 14. In this case, 000 no longer remains fixed, but is transformed into 001. Now 110 is transformed to itself and thus becomes fixed in this case.

The fixed word adder can add in succession any or all of the n bit words in whatever order is selected by the user.

Now referring to Figure 8, as an example, a block diagram of a system for carrying out encryption in accordance with the encryption and decryption techniques discussed so far may be seen. As shown in the figure, any value of the clear data 20, except 000, is provided as an address to memory 22. Stored at the various memory addresses is the order data for the clear data value, that is, the position, expressed as a binary number, of that clear data value in the ordered sequence of the right column of Figure 4 (and Figures 5 and 10). This position is provided as an output of the memory 22 to an adder shown as the combination of adders 24 and 26. The adders are coupled to add the output of the memory to a value of shift P_S as controlled by shift programmer 28. This addition is not a modulo 2 addition but rather is the normal binary add, with the one exception that the carry from the most significant bit is coupled to the carry in of the least significant bit. Thus, the adder will provide the result 001 as the sum 1 larger than 111, not 1000 or simply 000. Thus, it may be seen that the output of the adders is a new three bit binary number shifted in the order data sequence by an amount P_S. This new position is then used as the address for memory 30, which provides as its output the three bit binary number corresponding to the value in column 2 of Figure 4, or the corresponding clear data value in Figure 3. Thus, by way of example, if the clear data value is 010, that value as an address to memory I provides the location of that value of 001 in the sequence. If the shift program selects S = 6, then P_6 = 2 and column 3 is shifted downward two positions from column 2 or by an amount 010. The three bit binary number which would then be adjacent to the clear data value of 010 is 100 as in Figure 5. This added modulo 2 to the clear data 010 provides an encrypted value of 110, corresponding to the value shown in Figure 5. However, if the clear text data value is 000, that value as an address to Memory I provides the location of the value of 000 in the sequence. It is not shifted but provided unchanged as the order data in memory 30. Thus 000 added to itself, remains fixed.

The downward shift P_s of the sequence of column 3 of Figure 5 in comparison to the basic order data of column 2 of Figure 5 of course corresponds to a complimentary upward shift. Thus, for an n bit block, a downward shift of P_s is equivalent to an upward shift of $m-P_s$. Note also that for a three bit block, all values of possible shift provide the desired one-to-one mapping except for a shift of the first column with respect to the second column of zero, and of 7 and multiples thereof, as such shifts would provide a second column in the matrix having each row the same as the corresponding row of the first column, and any number added to itself modulo 2 will be zero. Thus, for a shift of seven or multiples thereof, all clear data values map to 000, useless for encryption purposes. In general however, it will be shown later that for n bit blocks larger than three bits, all shifts other than zero and integer multiples of m give the desired result and thus are usable in accordance with the original invention.

The block diagram for decryption in accordance with Figure 7 is shown in Figure 9. From a hardware standpoint, this diagram is exactly the same as that of Figure 8 for encryption, the decryption differing only in the shift S applicable for a given shift P_S for encryption. As in the example on page 14, for a shift P_S of 2 for encryption, a shift 6 provides the proper decryption, etc., as shown in the tables of Figures 8 and 9. Obviously, the encryption hardware and the decryption hardware must be using the associated shifts for the clear data to be properly recovered on decryption, though the applicable shift may be varied frequently at both ends to make cryptanalysis very difficult, if not virtually impossible.

If one adds modulo 2 a fixed number to any pair of columns of Figure 5, a still further one-to-one transformation results. By way of example, in Figure 10 the fixed number 001 has been added modulo 2 to the first and second columns of Figure 5. Now 010 as a clear text word maps into an encrypted word 111, whereas in the example of Figure 8, 010 mapped into 110.

An example of a block diagram for the encryption using a fixed word adder may be seen in Figure 13. This figure is identical to Figure 8 with the exception that the fixed word adder 32 has been included to add the fixed word (001 in the example) to the output of memory 30 corresponding to the value in the same row of the second column as 010 of the first column. Thus, the fixed word adder merely adds the fixed word (001 in the example) to the column 2 value, after which the clear text word is added modulo 2 thereto to obtain the encrypted data. Again for the example, using clear data of 010 as the address to memory 22, the output of the memory will be 001. Using the same shift as in the example of Figure 8, 010, $P_S = 2$ is added to the 001, to provide an address to memory 30 of 011. This results in an output from memory 30 of 100, to which fixed word adder adds modulo 2, the fixed word 001, yielding 101. This added modulo 2 to the clear text word 010 gives the encrypted word 111 as shown in Figure 10.

A block diagram for decryption, corresponding to the block diagram for encryption of Figure 13, is shown in Figure 14. As may be seen, Figure 14 is identical to Figure 13 (though the shifts for decryption are again different from the shifts for encryption), with the exception of the fixed word adder also adding modulo 2 the fixed word to the encrypted data before the same is applied to memory 22. This modulo 2 addition is in essence the second modulo 2 addition of the fixed word, as a first modulo 2 addition of the fixed word was done in Figure 11 to get the encrypted word. Thus, since a second modulo 2 addition of the same word in effect cancels the first modulo 2 addition so that after the encrypted data in Figure 12 has the fixed word added modulo 2 thereto, the result of that modulo 2 addition may be used with the equations of Figure 10 for decryption purposes. Thus, by way of example, using the encrypted word 111 of the example of Figure 13, 111 ⊕ 001 =110 as the address to memory 22 of Figure 14. This gives a memory output of 100, to which the value of S = 6 or 110 is added. 100 + 110 = 1010 => 011 with wrap-around. This in turn gives an address of 011 to memory 30 or an output thereof of 100, to which is added modulo 2 110, the address to memory 22, to recover the clear text data 010. Further of course, while the fixed word adder of Figures 13 and 14 used a fixed word 001, any other 3-bit fixed word may be used, or for that matter, the fixed word may be varied from time to time with or separate and apart from variations in the shift, a fixed word of 000 essentially reducing the operation of the system to that of Figures 8 and 9.

Obviously, the methods described in relation to Figures 6, 7, 11 and 12 may readily be carried out with a microprocessor based system under program control. Alternatively, the memory could readily be preprogrammed in read only memory used essentially as look-up tables, and the adders and modulo 2 adders could readily be conventional adder circuitry so that at least the major elements of an encryption and decryption system could be realized in either high speed discrete components or through a custom integrated chip. The shift program also could take various forms depending upon how often a shift is desired, the extent to which the shift order is itself varied, etc., microprocessor based, integrated circuits or other realizations being readily applicable, including shift register implementations as desired.

In Appendix 1 which follows, the transformations hereinbefore described are further analyzed and various properties and characteristics thereof are set forth. In Appendix 2, certain aspects of the method of block substitution of the parent application are reviewed, and the concepts of nonlinearity and nonlinear mappings of clear text to encrypted text (and vice versa) are presented. Nonlinearity in this sense means that the mappings of clear text to encrypted text (and from encrypted text to clear text) are nonlinear under the operation of bit-wise addition modulo 2. In that regard, it was pointed out that Figure 1 illustrates a many-one transformation of one set of three bit binary numbers to another set of binary numbers by a modulo 2 addition. This specific example maps the eight possible values of the three bit numbers in the first column by modulo 2 addition to six three bit numbers in column 3 representing six of the eight possible combinations, with two (100 and 011), each being repeated twice. Because two three bit numbers (010 and 101) map to the same three bit number (100), and two other three bit numbers (100 and 110) map to the same three bit number (010), the reverse mapping will have ambiguities, making the mapping illustrated in Figure 1 unsuitable for encryption and decryption purposes.

On the other hand, Figures 2 through 5 provide sets of equations for encryption of any of the eight possible three bit clear text words (column 1) to a corresponding non-ambiguous encrypted text word (column 3). These equations remain valid by the interchanging of columns 1 and 3, and thus with this interchange, form the equations for the corresponding decryption in the same way that the equations before the interchange form the equations for encryption. However, the set of equations shown in each of Figures 2 through 5 are linear in the sense that the addition of any two equations within a given set of equations (eight equations for three bit numbers such as in Figures 2 through 5) is also one of the equations of the set. For instance, in Figure 2, while the addition of the first or null equation to any other equation yields that other equation and is thus trivial, the addition of the second and third equations provides the fourth equation, the addition of the third and fourth equation provides the second equation, the addition of the fourth and fifth equation provides the eighth equation, etc. Even when one adds modulo 2 one equation to itself, one obtains one of the eight equations, namely the null equation, as may occur when one adds more than two equations modulo 2 such as, by way of example, equations two, three and four, as the addition of equations two and three yields equation four, and equation four added to itself yields the null equation. In that regard, adding two equations modulo 2 may be considered equivalent to adding any greater number of equations, as either or both of the equations added may be considered to be the sum of two or more other equations. Further, there is no combination of equations the sum of which is not another equation in the given set. What is most significant from a cryptanalysis standpoint is that given the right three of the seven equations other than the null equation, the remaining four equations may be determined by the appropriate sums of the combinations of the three known equations. For instance, while the combinations of sums of equations two, three and four of Figure 2 cannot be used to generate the rest of the equations, equations two, three or four, and five, six, seven or eight can be so used. Taking for example, equations two, four and eight, the sum of equations two and four provides equation three, the sum of equations two and eight provides equation seven, the sum of equations two, four and eight provides equation six, and the sum of equations four and eight provides equation five. Also the foregoing rule, of course, applies to encryption of words of other bit lengths, the generating equations for the sixteen equations for encryption of a four bit word being determined by adding modulo 2 various combinations of four independent equations.

With respect to the set of equations in Figure 10, adding any two equations does not provide a third equation of the set, though adding 001 to each of the left hand columns of the Figure 10 again provides the null equation and the rest of the set of equations of Figure 5, which set is generatable by any three independent equations of the set. It is this ability to generate the remainder of the equations from a basic set of independent equations which the present invention is intended to avoid, the present invention doing so in an orderly and logical manner so that not only may the base set of linear equations be varied from time to time or dynamically in the various ways disclosed in the original application, but the resulting base set may also be nonlinearized from time to time or dynamically to a varying extent and in varying combinations, making cryptanalysis much more difficult than before.

Referring again to Figure 2, if one rearranges the order of the equations, there is, of course, no change in the mapping of the numbers in column 1 to the numbers in column 3. Accordingly, the equations in Figure 2 may be rearranged as shown in Figure 15. In particular, it will be noted that, neglecting the null equation, the first number appearing in column 2 (001) occurs in the next line of column 1, the second number in column 2 (111) occurs in the third line of column 1, etc., the wraparound resulting in the last number in column 2 (101) falling on the first line of column 1 (again neglecting the null equation). The resulting organization of the equations is in the form illustrated on page 7 of Appendix 2, where in Figure 15, x_1 is 001 and x_m is 101. Any set of equations for words (numbers) of any bit length having a null equation and $2^n - 1$ non-zero equations may be so arranged without any changing of the mapping defined thereby, as such an arrangement is a mere changing of the order of appearance of the equations and not a changing of any of the equations themselves.

It is shown in Section 3.2 of Appendix 2 that certain groups of such equations may be altered by rearrangement of the words appearing in columns 1 and 2 to provide correspondingly new modulo 2 addition equations, which when substituted for the original group of equations within the original set of equations still maintains a one to one mapping and thus is suitable for use in encryption and decryption. In that regard, the one to one mapping is preserved because the order of the multibit words appearing in columns 1 and 2 of the selected group of equations is changed, but not the words themselves, so that the group of words mapped and the group of words to which they are mapped by the selected equations has not been changed, though within those two groups, which word in column 1 maps to which word in column 3 has been changed. The net effect of these changed equations is that the same no longer are linear extensions of the unchanged equations, that is, the same can no longer be generated by the addition of two or more of the unchanged equations. This, therefore, breaks up the linearity of the original set, the possible extent of which will be subsequently discussed, making the cryptanalysis more difficult as desired.

It is shown in Section 3.2 of Appendix 2 that under certain conditions, groups of equations within a given set may be altered and used to replace the corresponding original group of equations within the original set so as to maintain a one to one mapping for the complete set, and at the same time break up the linear characteristic of the set of equations as hereinbefore described. These conditions are more specifically illustrated in equation form in Section 3.3 of Appendix 2, wherein the two possible modifications are illustrated in equation form. The basic concept is to take sums of consecutive triples of rows in the original set of equations, with the analysis in Section 3.2 of Appendix 2 showing that, as stated in Section 3.3, the nonlinearization by taking such consecutive triples of rows works if, and only if, a set of only three or four consecutive rows of the original set are used. If three consecutive rows are used, four rows are actually modified, namely the three consecutive rows of the original set, together with a fourth row corresponding to the vector sum modulo 2 of the three consecutive rows. The modification can be obtained by adding vectorially to each of the four rows, the following equation:

$$(x_1 \oplus x_2) \oplus (x_1 \oplus x_2) = \Theta$$

If four consecutive rows of the original set of linear equations are used, six rows of the original set of equations are modified, namely the four consecutive rows, together with the row representing the vector sum of the first three of the four consecutive rows, and the row corresponding to the vector sum of the last three of the four consecutive rows of the original set (e.g. the row corresponding to the sum of rows 1, 2 and 3, and the row corresponding to the sum of rows 2, 3 and 4, as shown on page 10 of Appendix 2). The modification in this case may be obtained by adding vectorially to the corresponding six rows the following:

to rows 1 and q $(x_1 \oplus x_2) \oplus (x_1 \oplus x_2) = \Theta$

to rows 2 and 3 $(x_1 \oplus x_3) \oplus (x_1 \oplus x_3) = \Theta$

to rows 4 and q + 1 $(x_2 \oplus x_3) \oplus (x_2 \oplus x_3) = \Theta$

The form of the equations above and the original equations shown on page 10 of page A2 of Appendix 2 suggests that nonlinearization works if one takes the first, second, third and one other row of the original set of linear equations, or alternatively, if one takes the first, second, third, fourth and two other rows of the original set of linear equations. Since the method works because the equations in the original set as selected for modification are linear within themselves, equations once nonlinearized by the methods of the present invention may not be again used as part of the nonlinearization process. This would tend to suggest that only four or six equations could be nonlinearized by this process, which of course would be an insignificant number of the total equations for larger word sizes (for instance, a four bit word requires 16 equations, an eight bit word 256 equations, etc.). However, again referring to Figure 15, it is to be noted that which word or number in column 2 is to be selected from the non-null rows as x_1 is arbitrary. By way of example, if one selected 011 as x_1 rather than 001, the third non-zero line would become the first, the fourth non-zero line the second, the fifth non-zero line the third, the sixth non-zero line the fourth, the seventh non-zero line the fifth, the first non-zero line the sixth, etc., essentially shifting the lower five equations up and wrapping the upper two non-zero equations around, with the result that the equations themselves are not changed, nor is the ordering of the equations, but rather only the starting point in that sequence is changed. Such an arrangement of equations was shown in Figure 3, wherein x_1 = 100 and x_m (= x_7) = 011. Thus the equations presented on page 10 of Appendix 2 are general in the sense that if three consecutive rows and the row corresponding to the sum of the three consecutive rows are to be modified (nonlinearized) any three consecutive rows may be so selected, limited only by the fact that none of the three selected nor the row corresponding to the sum of the three can have previously been nonlinearized as a result of an earlier selection. Similarly, if four consecutive rows plus the two sum rows hereinbefore described are selected, any four consecutive rows may be so used, again provided that none of the four selected nor of the two sum rows may have previously been nonlinearized by this process. To generalize the equations for nonlinearization, one need only consider x_1 as being the value in the second column of the first of the three or four successive rows selected, and renumbering values in each column accordingly.

It will be noted that the nonlinearization process is carried out on the equations other than the null equations. Since there are $2^n - 1$ such equations, wherein n is the bit length of the word used, there is necessarily an odd number of equations available for nonlinearization regardless of the value of n, whereas the nonlinearization process nonlinearizes an even number (4 or 6) equations at a time (obviously in a high speed system, apparatus may be provided to simultaneously nonlinearize different non-overlapping groups of a given set of linear equations, as the nonlinearization processes for non-overlapping groups are totally mutually independent, regardless of which process is used). Thus, it is clear that not all equations in any given linear set may be nonlinearized. Consequently, there is a question as to how many of the equations may be nonlinearized, and whether there is a logical manner of selecting equations for nonlinearization. These considerations are discussed in Sections 3.4 through 3.6 of Appendix 2. In general, while not all equations may be nonlinearized, normally a vast majority of the equations may be nonlinearized for word sizes of four or more bits, leaving the remaining nonlinearized equations of little significance, and perhaps if anything, possibly misleading from a cryptanalysis viewpoint. Further of course, it should be noted that varying from time to time or dynamically varying the number and identification of the rows to be nonlinearized and which nonlinearization technique is used further compounds the cryptanalysis problem, though such time varying or dynamically varying nonlinearization is not that difficult from a hardware standpoint (or software standpoint, if done under software control) as the starting set of linear equations (which themselves may be varied from time to time or dynamically, as herein before described) may be generated from a simple and readily variable generating function, which set of equations may be nonlinearized in both manner and extent utilizing logical processes, which manner and extent may each themselves be varied from time to time or dynamically.

As an example of the foregoing, attention is directed to the table on page A1 (Appendix A of Appendix 2 hereof) which provides the sixteen equations for the linear mapping of a four bit number or word to another four bit number or word utilizing a specific generating function. Note that these sixteen equations are organized in the manner indicated for the original equations on page 10 of Appendix 2. As noted on page A1, it is easily verified that the sum of any two of the sixteen equations on page A1 is another of the sixteen equations in accordance with the concept of linearity as used herein. This table on page A1 is nonlinearized as described on page A8 and is presented in its nonlinearized form on page A9 of Appendix 2. In particular, the nonlinearization is in accordance with the first method, namely, utilizing three consecutive rows of the original set of equations (neglecting the null equation), plus the row representing the sum of the first three rows. In that regard, the sum modulo 2 of the first three non-zero numbers in column 1 (1001, 0001 and 0010) is equal to 1010, the value in the eleventh row of the non-zero equations. Thus rows one, two, three and eleven are nonlinearized by adding modulo 2 $x_1 \oplus x_2$ to each of columns 1 and 2 thereof. To be more specific, x_1 equals 0001 and x_2 equals 0010, so that $x_1 \oplus x_2 = 0011$. Adding modulo 2 0011 to the first equation gives $1010 \oplus 0010 = 1000$ (1000 is the original value in column 3 for the first equation) as shown in the table on page A9. The same addition for the equations on lines 2, 3 and 11 carries out the transformation for these four lines. Similarly, if one adds lines 5, 6 and 7 of the non-zero equations, one obtains the equation of line 15 of the non-zero equations, the last non-zero equation shown on page A1. These four lines may be nonlinearized in the same manner as lines one, two, three and eleven, noting however that the applicable equation is effectively now:

$$(x_5 \oplus x_6) \oplus (x_5 \oplus x_6) = \Theta$$

With respect to further nonlinearization of the set of sixteen equations on page A9 of Appendix 2, there are two other series of three consecutive equations in the table, specifically, lines 8, 9 and 10 and 12, 13 and 14 which might be considered. The modulo 2 sum of lines 8, 9 and 10 however, provide line 3 of the non-zero equations, a line already used, and the modulo 2 sum of lines 12, 13 and 14 provide line 7, another line already used. Accordingly, while two additional groups of three consecutive lines or three consecutive equations exist, the same cannot be used for further nonlinearization because the sum of either of the three is a line or equation which has already been nonlinearized.

As another example, note the table set out at the top of page A4 of Appendix 2 hereof. This set of linear equations uses the same generating function but as applied to a new base (see the bottom of page A3 of Appendix 2), which when nonlinearized using the same set of equations as in the previous example (equations 1, 2, 3, 5, 6, 7, 11 and 15) provides the nonlinear set of equations set forth on page A11 of Appendix 2.

Finally, as a third example, note the example described near the bottom of page A11, with the nonlinearized equations shown on page A12. This example is an example of another nonlinearization of the table of 15 equations (together with the null equation) presented on page A1 of Appendix 2, nonlinearized using a different basis, specifically four successive (non-zero) equations 1, 2, 3 and 4 together with the sum of 1, 2 and 3, namely equation 11, and the sum of equations 2, 3 and 4, namely equation 12, together with the three successive equations 13, 14 and 15 and the sum thereof, equation 8. The equations for nonlinearizing four consecutive equations plus the two modulo 2 sum equations of course have been given before herein and are set out on page 10 of Appendix 2. In particular, three different equations are used, one for rows 1 and q, one for rows 2 and 3, and one for rows 4 and q + 1. By way of example, taking row 1, zero is added to column 3 and $x_1 \oplus x_2$ is added modulo 2 to each of columns 1 and 2 (the modulo 2 sum of anything to itself equalling zero). Since $x_1 \oplus x_2 = 0011$, adding this to equation 1 yields the equation $1010 \oplus 0010 = 1000$, the first non-zero equation in the nonlinearized set of equations on page A12 of Appendix 2. For row 2 of the linear set of non-zero equations, $x_1 \oplus x_3$ is added to each of columns 1 and 2, namely $0001 \oplus 0100 = 0101$. Adding this to columns 1 and 2 of row 2 of the linear set of equations of page A1 yields the fifth non-zero equation in the set of equations on page A12.

Finally, as an example of the use of the third equation for rows 4 and q + 1, $x_2 \oplus x_3 = 0010 \oplus 0100 = 0110$. Adding this, for example, to columns 1 and 2 of row 4 of the linear non-zero equations yields row 2 of the non-zero equations in the nonlinearized set of equations on page A12. Of course all six of the applicable rows must be modified in accordance with the nonlinearization process. Thus, in this latter example, 10 of the equations are nonlinearized instead of the 8 in the prior example, and of course the resulting mapping from column 1 to column 3 is generally quite different for the two sets of equations.

Finally, the nonlinearized equations may be further modified by adding modulo 2 an offset to each of the first two columns. This, of course, is equivalent to adding the offset modulo 2 to itself which of course is 0 and therefore does not affect the numbers in the third column. By way of specific example, Figure 17 presents the set of equations of the third example described above and shown on page A12 of Appendix 2 as modified by the addition of the offset 0101 to the first and second columns.

Figure 16 shows a block diagram of typical apparatus for encryption and decryption in accordance with the present invention. As may be seen in Figure 16, it is convenient to ultimately use a look-up table in the form of a read/write memory wherein the clear text data block or the encrypted text data block (both n bits long) is presented in parallel as the address to the memory with the data stored at the corresponding address corresponding to the encryption or decryption of the respective data block, respectively. For that purpose, it may be convenient to use a memory of twice the address space of that required for either encryption or decryption (e.g. n + 1 address bits) so that the memory address range is one bit wider than the data block to be operated on. In this manner, one bit of the memory address may be used to designate whether the operation is to be an encryption or a decryption operation. By way a specific example, the most significant bit of the memory address might be 0 to indicate a decryption process or a 1 to indicate an encryption process, with the decryption data stored in the lower half of the address range of the memory and the encryption data stored in the upper address range of the memory. Thus both encrytion and decryption may be done as desired by the look-up table by control of the single bit, and encryption or decryption of a block of n bits may be achieved in a single memory cycle.

Assuming that the mappings for encryption and decryption are to be changed periodically and/or dynamically, some method of altering the contents of the look-up table must be provided. While this could be done by specialized hardware, it is convenient to do the same by an appropriate processor under program control, as the alteration of the encryption and decryption schemes normally will occur far less frequently than the encryption and decryption process itself must be carried out. Accordingly, the same normally need not be accomplished with the same speed as encryption and decryption itself. Accordingly, the nonlinear dynamic substitution generator shown in Figure 16 may operate under program control based on various inputs thereto. In particular, the equation for encryption may readily be generated under program control given certain basic information defining the same, such as by way of example the block substitution bit size (n), the base set of n linearly independent numbers, the generating function, the beginning equation of the linear set on which to begin nonlinearizing, and the number of iterations of the nonlinearizing function to perform.

Once the offset has been applied to the nonlinearized equations, each number or block in column 3 is stored in the portion of the look-up table assigned to encryption at an address equal to the block in column 1 for the respective row. Thus, when a number or block in column 1 is applied as the address, the number read out of the memory is the number in column 3 for that row representing the respective encrypted block. For the decryption portion of the table, the process is reversed, in that the blocks in column 3 are used as memory addresses (more appropriately address portions, the full address including the address bit designating decryption) with the data stored at those addresses being the respective blocks in column 1. Thus, during decryption the memory is entered at the address defined by the encrypted block, with the data stored at the respective address being provided as the output corresponding to the associated clear text block. For convenience, detailed methods for encrytion and decryption are set out in Appendix 3.

Obviously the encryption and decryption processes could be carried out entirely under program control, as both processes simply involve logical manipulations given certain (variable) starting information. However, the speed with which encryption and decryption could be carried out would be very grossly reduced, as the processor would wind up regenerating the same encryption and decryption equations over and over again. In comparison, the use of the look-up table allows a one time determination of the full set of encryption and decryption equations, which information for any data block to be encrypted or de-encrypted is continuously available in a single memory cycle until such time as the equations are to be changed.

While a preferred embodiment for the encryption and decryption of the present invention has been disclosed and described herein, it will be obvious to one skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

APPENDIX 1

UNBIASED BLOCK SUBSTITUTIONS

ABSTRACT

A block substitution is a one-to-one mapping of the n-bit binary numbers onto themselves. Such a substitution or transformation can be represented by a permutation. It is shown that certain permutations of the n-bit binary numbers define a block substitution by modulo 2 addition of one permuted set of numbers to another. Such permutations are termed replicative. A subset of these have an additional feature which is that the equations which they define have an additive relationship when viewed as vectors. Such permutations are termed additive. An unbiased block substitution is defined. It is shown that substitutions defined by an additive permutation are unbiased and that any unbiased block substitution can be represented by a replicative permutation. Additive permutations are shown to form groups which retain the same properties. The conditions for existence of these additive permutations are established, some properties of the groups determined, and the number of such groups enumerated and compared with all possible permutations of the n-bit numbers.

INTRODUCTION

A block substitution is the term usually applied to a one-to-one mapping of the n-bit binary numbers onto themselves. This mapping can be written as a pairing of the 2^n n-bit numbers:

$$\begin{array}{ccc} x_1 & & z_1 \\ x_2 & & z_2 \\ \cdot & & \cdot \\ \cdot & & \cdot \\ x_k & \longrightarrow & z_k \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \end{array}$$

where each column is the set of the same 2^n distinct n-bit numbers but written in different orders. Thus, this mapping can be thought of as a permutation of the n-bit numbers written as:

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_k & \cdots \\ z_1 & z_2 & \cdots & z_k & \cdots \end{pmatrix}$$

or $(x_1\ x_i\ x_j\ \ldots)$ for some set of indices. This usual notation for permutations simply means that $x_1 \longrightarrow x_i$, $x_i \longrightarrow x_j$, etc.

Going back to the column notation, one could define a set of simple equations from the original set and its image:

$$\begin{array}{ccccc} y_1 & \oplus & x_1 & = & z_1 \\ y_2 & \oplus & x_2 & = & z_2 \\ & \cdot & & & \cdot \\ & \cdot & & & \cdot \\ & \cdot & & & \cdot \\ y_k & \oplus & x_k & = & z_k \\ & \cdot & & & \cdot \\ & \cdot & & & \cdot \\ & \cdot & & & \cdot \end{array}$$

where $\oplus$ means modulo 2 addition (i.e., addition of corresponding digits without carrying). In general, the set $\{y_1, y_2 \ldots\}$ will not all be distinct but in certain circumstances they will be. When they are distinct, block substitutions can be generated by modulo 2 addition rather than by conventional means. The main tasks are to determine the circumstances, if any, in which this scheme works, how the substitutions can be quickly changed, and the lack of bias.

It is not obvious that block substitutions can ever be generated by modulo 2 addition. For example, consider the attempt to substitute one arrangement of 3-bit binary numbers for another by modulo 2 addition. This is shown in Figure 1. In column 3, on the right, 011 and 100 each appear twice, while 001 and 110 never appear. The numbers in column 1, on the left, acting on the numbers in column 2, in the center, constitute a transformation of the set of 3-bit binary words into themselves. This is a many-one transformation and is useless for block substitutions.

$$\begin{array}{ccccc}
000 & \oplus & 000 & = & 000 \\
001 & \oplus & 011 & = & 010 \\
010 & \oplus & 110 & = & 100 \\
011 & \oplus & 100 & = & 111 \\
100 & \oplus & 111 & = & 011 \\
101 & \oplus & 001 & = & 100 \\
110 & \oplus & 101 & = & 011 \\
111 & \oplus & 010 & = & 101
\end{array}$$

Figure 1. Modulo 2 Addition, Many-One Transformation

Trying another arrangement shown in Figure 2 gives a different result. In this case, the transformation is one-to-one from the 3-bit binary numbers onto themselves. Each column consists of all the 3-bit numbers exactly once.

$$\begin{array}{ccccc}
000 & \oplus & 000 & = & 000 \\
001 & \oplus & 111 & = & 110 \\
010 & \oplus & 011 & = & 001 \\
011 & \oplus & 100 & = & 111 \\
100 & \oplus & 110 & = & 010 \\
101 & \oplus & 001 & = & 100 \\
110 & \oplus & 101 & = & 011 \\
111 & \oplus & 010 & = & 101
\end{array}$$

Figure 2. Modulo 2 Addition, One-to-One Transformation

Definition: A replicative array of n-bit binary numbers is a set of $2^n = m+1$ equations:

$$\begin{array}{c}
y_1 \oplus x_1 = z_1 \\
\cdot \\
\cdot \\
\cdot \\
y_{m+1} \oplus x_{m+1} = z_{m+1}
\end{array}$$

where the sets $\{y_k\}$, $\{x_k\}$, and $\{z_k\}$ each consist of the 2^n distinct n-bit binary numbers.

The set $\{y_k\}$ defines the mapping $x_k \longrightarrow z_k$. Since the y_k take on all values, exactly one member of the set $y_j = I = (0\cdots0)$ the identity. Thus, $x_j = z_j$ is a fixed point and there is no other. Since each column consists of the distinct n-bit numbers, it is a permutation of each of the other two columns.

The first question is how to construct such a replicative array.

Proposition 1: A replicative array of n-bit binary numbers can be constructed from sums of any set of n+1 rows each of whose columns contain the identity and n generators of G_n, the group of n-bit binary numbers.

Proof: Without loss of generality we can assume that the first n+1 rows have this property. Either one row is $I = I \oplus I$ or the identity occurs in three different rows. Assume $z_1 = x_1 = y_1 = I$. Then each set $\{z_2, \ldots, z_{n+1}\}$, $\{x_2, \ldots, x_{n+1}\}$, $\{y_2, \ldots, y_{n+1}\}$ is composed of n distinct generators of G_n. To construct the remaining rows of the array, take modulo 2 sums of pairs of rows 2 through n+1, sums of triples of these rows, etc, and, finally, $\sum_{i=2}^{n+1} z_i$. The number of additional rows constructed in this manner is:

$$\binom{n}{2}+\binom{n}{3}+\ldots+\binom{n}{n} = \sum_{k=0}^{n}\binom{n}{k}-\binom{n}{1}-\binom{n}{0} = 2^n - (n+1).$$

So the array is completely specified by this process. Since each of the z_i, x_i, and y_i for $i > n+1$ are different linear combinations of the generators, they will all be distinct, and the sets of 2^n elements of $\{z_i\}$, $\{x_i\}$ and $\{y_i\}$ will each be a permutation of the n-digit binary numbers.

Now assume that $z_1 = x_2 = y_3 = I$. No pair of rows from the first three could be added together because this would cause a duplication of z_1, x_2, or y_3 in one of the columns. We avoid this problem by adding together odd numbers of rows from the initial n+1 rows. Thus we have:

$$n+1 = \frac{(n+1)!}{(n+1)!1!} = \binom{n+1}{1} \quad \text{single rows}$$

$$\frac{(n+1)!}{3!(n-2)!} = \binom{n+1}{3} \quad \text{triple sums of rows}$$

$$\frac{(n+1)!}{5!(n-4)!} = \binom{n+1}{5} \quad \text{quintuple sums, etc}$$

Keeping in mind the general identity for binomial coefficients:

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$$

we can sum the number of rows generated this way, for n odd:

$$\binom{n+1}{1} + \binom{n+1}{3} + \ldots + \binom{n+1}{n} = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \binom{n}{3} + \ldots + \binom{n}{n-1} + \binom{n}{n} = 2^n$$

For n even, the final term in the sum on the left is:

$$\binom{n+1}{n+1} = \binom{n}{n}$$

so that it again adds to 2^n rows.

As in the first case, the column elements are 2^n distinct linear combinations of the generators and the identity. (Q.E.D.)

Proposition 1 gives a method of generating replicative arrays but does not imply that all can be generated this way. It is convenient to introduce another definition. The individual rows of the replicative array can be thought of as vectors which can be added to each other by adding the corresponding scalar components modulo 2.

Definition: A replicative array, which has the property that the vector sum modulo 2 of any odd number of rows is again another row in the original array, will be called an additive array.

Clearly, replicative arrays generated as in proposition 1 are additive arrays, but there are others as well which do not have this additive property.

Proposition 2: Any additive array of n-bit numbers with fixed row $I \oplus I = I$ has a maximal set of n linearly independent rows.

Proof: Let k be the maximal number of independent rows and assume $k < n$. The additive array has exactly 2^n rows including the identity. By proposition 1 these k rows generate an array of 2^k rows. Thus, the remaining $2^n - 2^k$ rows must be duplicates, and the array cannot be additive.

If $k = n+q$ where q is a positive integer, then a maximal set of independent rows will generate 2^{n+q} rows. In each column, each of the 2^n distinct n-bit numbers will appear 2^q times. This 2^n additive array is embedded in the array with 2^{n+q} rows. One can extract it by making a proper selection of rows from the larger array. Choose the first row for the additive array, for example, $x_1 \oplus y_1 = z_1$. x_1 will appear 2^q-1 more times in column 1, similarly for y_1 in column 2 and z_1 in column 3. Since the same combination of x_1 and y_1 will not occur again, 3 (2^q-1) rows must be eliminated from further consideration for the additive array. From the second choice, z_2, one must eliminate 2^q-1 rows containing z_2 in column 3, but since x_2 or y_2 may have appeared in a row already eliminated, we can only say that in the second step, the number of rows eliminated is $\geq 2^q-1$. For each succeeding step $\geq 2^q-1$ rows are eliminated. The process is complete after 2^n steps. 2^n rows have been selected and the number of rows eliminated is:

$$\geq 3(2^q-1) + (2^n-1)\,(2^q-1) = (2^n+2)\,(2^q-1)$$

Thus, the total number of rows selected and eliminated is:

$$\geq 2^n + (2^n+2)\,(2^q-1) = 2^{n+q} + 2(2^q-1) > 2^{n+q}.$$

So there are not sufficient rows to complete this process. This contradiction implies $k \neq n$. (Q.E.D.)

The above may seem to be rather academic, but it shows how to generate additive arrays and the resulting substitutions, and also shows that this process gets them all. But these additive arrays have another very useful property described in the next proposition.

Proposition 3: An additive array with fixed identity row is a group.

Proof: Rows in the array can be thought of as vectors and added by respective components:

$$\begin{array}{cccc} x_i \oplus & y_i & = & z_i \\ x_j \oplus & y_j & = & z_j \\ \hline x_k \oplus & y_k & = & z_k \end{array}$$

where $x_i \oplus x_j = x_k$, etc. This is the group operation. Each row in the array is the sum of some subset of the n generators. The sum of two rows is also the sum of some subset of the generators and, thus, is in the array which consists of all possible such sums. The group identity is the row $I = I \oplus I$. (Q.E.D.)

Each pair of columns defines a permutation as follows: take a number in one column, e.g., y_1 in column 2, and find the same number in column 1. It will have a partner, say y_2, in column 2. Again, locating y_2 in column 1, there will be a partner y_3 in column 2. This defines a permutation $y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow \cdots$ from column 2 to column 1. Six such permutations, three of which are distinct, are defined by the additive array.

Definition: A permutation defined by an additive array will be called an additive permutation.

Initially, consider the case where the fixed point is I, and let $y_{m+1} = x_{m+1} = z_{m+1} = I$. The order of rows in the additive array is arbitrary. For the second row, one could select $y_j = x_1$ and for the third row, $y_k = x_j$, etc. Redesignating the indices, one obtains:

$$\begin{array}{ccccc} I & \oplus & I & = & I \\ x_m & \oplus & x_1 & = & z_1 \\ x_1 & \oplus & x_2 & = & z_2 \\ x_2 & \oplus & x_3 & = & z_3 \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \end{array}$$

The order of the numbers in columns 1 and 2 now represents the permutation defined by those two columns which leads to:

Definition: *An additive array whose rows are arranged in the order* defined by the permutation between columns 1 and 2 will be said to be normalized.

Proposition 4: An additive array with fixed identity can be written in the form:

$$\begin{array}{cccc}
I & \oplus & I & = & I \\
x_m & \oplus & x_1 & = & x_{1-p} \\
x_1 & \oplus & x_2 & = & x_{2-p} \\
 & & \cdot & & \\
 & & \cdot & & \\
 & & \cdot & & \\
x_{m-1} & \oplus & x_m & = & x_{m-p}
\end{array}$$

where p is an integer.

Proof: In normalized form the array can be written:

$$\begin{array}{ccccc}
I & \oplus & I & = & I \\
x_m & \oplus & x_1 & = & z_1 \\
x_1 & \oplus & x_2 & = & z_2 \\
 & & \cdot & & \\
 & & \cdot & & \\
 & & \cdot & & \\
x_{m-1} & \oplus & x_m & = & z_m
\end{array}$$

Since the array is a group, it can be regenerated by taking sums of successive pairs of rows to obtain:

$$\begin{array}{ccccccccccc}
 & & & I & & \oplus & & I & = & I & \\
(x_1 & \oplus & x_m) & \oplus & (x_1 & \oplus & x_2) & = & (z_1 & \oplus & z_2) \\
(x_1 & \oplus & x_2) & \oplus & (x_2 & \oplus & x_3) & = & (z_2 & \oplus & x_2) \\
 & & & & & \cdot & & & & & \\
 & & & & & \cdot & & & & & \\
 & & & & & \cdot & & & & & \\
(x_{m-1} & \oplus & x_m) & \oplus & (x_m & \oplus & x_1) & = & (z_m & \oplus & z_1)
\end{array}$$

The rows in the new array must be the same (but rearranged) as those in the original. From the diagonal structure on the left, the order of the rows must be the same although the starting point in the sequence of rows has been shifted by some unspecified number of positions. Column 2 in the new array is:

$$\begin{array}{c}
z_2 \\
z_3 \\
\cdot \\
\cdot \\
\cdot \\
z_1
\end{array}$$

which is the same as column 3 in the original array. So column 3 in the original array is the same as column 2 in the original array but shifted by some unspecified number of positions p. (Q.E.D.)

As will be seen later, the shift p can take on only certain values. So far, no means has been found to determine p in advance or from a knowledge of p, to determine the array, except for special cases. However, there are some rules for eliminating possible values of p and for pairing permissible values. The shift p determines whole classes of arrays and substitutions. So far, all possible values of p have been found for block size $n \leq 8$.

Proposition 5: If g is a maximal length additive permutation of n-bit binary numbers with fixed identity, then for any power s of the permutation g, g^s is also additive.

Proof: g is defined by columns 1 and 2 of the following additive array, omitting the identity:

$$\begin{array}{ccccc} x_m & \oplus & x_1 & = & x_{1-p} \\ x_1 & \oplus & x_2 & = & x_{2-p} \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ x_{m-1} & \oplus & x_m & = & x_{m-p} \end{array}$$

The array which defines g^s is:

$$\begin{array}{ccccc} x_{1-s} & \oplus & x_1 & = & w_1 \\ x_{2-s} & \oplus & x_2 & = & w_2 \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ x_{m-s} & \oplus & x_m & = & w_m \end{array}$$

As in the proof of proposition 4, from the group structure of the array for g, g can be regenerated by taking the sums of the m pairs of rows which are s spaces apart:

$$\begin{array}{rcl} (x_m \oplus x_s) \oplus (x_1 \oplus x_{s+1}) & = & w_1 \oplus w_{s+1} \\ (x_1 \oplus x_{s+1}) \oplus (x_2 \oplus x_{s+2}) & = & w_s \oplus w_{s+2} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ (x_{m-1} \oplus x_{s+m-1}) \oplus (x_m \oplus x_{s+m}) & = & w_m \oplus w_{s+m} \end{array}$$

From the diagonal structure on the left, it is clear that columns 1 and 2 are in the same order as before but with a different starting point, i.e., the array has been rotated. Noting that $x_{s-s} = x_m$, from the array for g^s, $x_m \oplus x_s = w_s$ Thus, column 1 (or column 2) of the rotated array for g consists of the set $\{ w_1, \ldots, w_m \}$ which is a displaced or rotated form of $\{ x_1, \ldots, x_m \}$. (Q.E.D.)

This shows that the general form of a row in the additive array for g^s is:

$$x_{k-s} \oplus x_k = x_{k-p_s}$$

where p_s is the shift corresponding to the power s. p_s is related to the basic shift p and is different for each value of s. It is important to know these shifts and their sequences when changing from one substitution to another in the same permutation group. More will be said about this later.

In proposition 5, the property of being additive is necessary. Examples can be shown of replicative permutations g which are not additive and for which g^s is not replicative.

Proposition 5 is also true even if g is not a maximal length permutation. The presence of cycles slightly complicates the proof. However, there is little practical interest in starting with substitutions corresponding to non-maximal permutations because powers will not generate the full group. The permutations which have fixed points other than the identity also form groups. However, the additive arrays which generate them are not groups.

The next step is to show that these groups of permutations and the corresponding substitutions have the highly desirable property of yielding all possible input/output pairs.

Since each additive permutation is generated by an array in which pairs of columns keep exactly one element fixed, each permutation will keep one of the 2^n binary words fixed. Thus, these permutations not only belong to the symmetric group of all permutations on 2^n elements, but they also belong to the subgroup P_m of permutations on $m = 2^n-1$ elements. All powers of a given permutation will form a group. The order of the permutation is the least common multiple of its cycles, and the order of the group is the number of distinct permutations in it. A permutation is said to be regular if all its cycles are the same length.

An important property for a group of permutations is transitivity. A transitive group has at least one permutation which transforms any element or word into any other.

Proposition 6: If G is a subgroup of P_m which is generated by powers of an additive permutation g, where g is of order $m = 2^n-1$, then G is transitive on the 2^n-1 non-fixed words.

Proof: Omitting the single cycle permutation (j) corresponding to the fixed word, x_j, G is a subgroup of permutations on the 2^n-1 n-bit words that are held fixed. Since the m powers of g $\{g, g^2, g^3, \ldots\; g^m = I\}$ generate G, its order $|G| = m$. Let G_1 be the subgroup of G which holds fixed one of the 2^n-1 n-bit words. Since this is a group of additive permutations, $G_1 = \{I\}$ the identity permutation only. So G_1 has order $|G_1| = 1$ and the index of G_1 in G $\left|\frac{G}{G_1}\right| = m$. So G is transitive. (Q.E.D.)

ADDITIVE ARRAYS WITH OTHER FIXED POINTS

One can generate additive arrays with a fixed element other than the identity by simply adding a fixed binary number to each number in column 1 and column 2. For example, let H be the array (and group) as written in proposition 4, and generate a new array H' as follows:

$$\begin{array}{ccccc}
 & & H & & \\
I & \oplus & I & = & I \\
x_m & \oplus & x_1 & = & x_{1-p} \\
x_1 & \oplus & x_2 & = & x_{2-p} \\
 & & \cdot & & \\
 & & \cdot & & \\
 & & \cdot & & \\
x_{m-1} & \oplus & x_m & = & x_{m-p}
\end{array}
\qquad
\begin{array}{ccccc}
 & & H' & & \\
w & \oplus & w & = & I \\
(x_m \oplus w) & \oplus & (x_1 \oplus w) & = & x_{1-p} \\
(x_1 \oplus w) & \oplus & (x_2 \oplus w) & = & x_{2-p} \\
 & & \cdot & & \\
 & & \cdot & & \\
 & & \cdot & & \\
(x_{m-1} \oplus w) & \oplus & (x_m \oplus w) & = & x_{m-p}
\end{array}$$

Clearly, H' has 2^n rows and the sum of any pair or even number of rows will be in H since the w's will cancel out. If any odd number of rows in H' are summed, it will result in a row in H' because it is equivalent to the sum of rows in H plus the row $w \oplus w = I$. Thus, $H \cup H'$ has 2^{n+1} elements, is closed under addition of rows, and contains the identity $I \oplus I = I$. Thus, $H \cup H'$ is a group, H is a maximal subgroup, and $H' = \overline{H}$ is the relative complement of H.

We can shift column 1 in H' to obtain the array corresponding to a power of the permutation defined by the original array. A typical row is:

$$(x_{k-j} \oplus w) \oplus (x_k \oplus w) = x_k - p_j$$

This is the image of the corresponding row in the jth power of H. So the permutation defined by H' is additive and so are its powers.

Now, suppose that we have an additive array with 2^n rows for which I is not fixed. Further, assume that the array contains sums of all odd combinations of $n+1$ linearly independent rows. As in the proof of proposition 1, this generates exactly 2^n rows. Thus, no sums of even sets or rows can be in the array. Since I occurs once in column 3, there is one row of the form $w \oplus w = I$. Adding it to each of the single rows, including itself, generates an additive array of 2^n rows with fixed identity.

Thus, there is a one-to-one correspondence between additive arrays with fixed identity and additive arrays with another fixed number.

Thus, all the additive arrays of interest for use as block substitutions have either fixed identity or can be derived from such arrays as described above. In what follows, additive arrays will have fixed identity unless otherwise specified. Typically, the row $I \oplus I = I$ will be omitted.

GENERATION OF ADDITIVE ARRAYS

It is helpful to know something about the distribution of independent rows or generators in the array.

Proposition 7: In a maximal additive array with fixed I, on the n-bit binary numbers, any n consecutive rows are generators.

Proof: Consider the first n rows:

$$x_m \oplus x_1 = x_{1-p}$$
$$\vdots$$
$$x_{n-1} \oplus x_n = x_{n-p}$$

If these are not independent then

$$x_n = \sum_{i \in Q} x_i \qquad x_{n-1} = \sum_{i \in Q} x_{i-1} \qquad x_{n-p} = \sum_{i \in Q} x_{i-p}$$

where Q is a subset of $\{1, \ldots., n-1\}$ with $s \leq n-1$ elements. The next row in tne array will be: $x_n \oplus x_{n+1} = x_{n+1-p}$. Since the array is an additive group, it must be true that $x_{n+1} = \sum_{i \in Q} x_{i+1}$ and $x_{n+1-p} = \sum_{i \in Q} x_{i+1-p}$. In that case applying this to each successive row, one could generate the entire array witn $s < n$ generators. (Q.E.D.)

This is enough for our purposes, but it can also be shown that any n equally spaced rows are generators.

THE SHIFT p

In what follows, it is assumed, unless otherwise stated, that the additive arrays are maximal, normalized, and with fixed identity. For brevity, the row I $\oplus$ I = I will be omitted. The general form for the kth row of the basic array is:

$$x_{k-1} \oplus x_k = x_{k-p}$$

For the array corresponding to the power s of the basic permutation, the kth row is:

$$x_{k-s} \oplus x_k = x_{k-p_s}$$

As usual, n is the block size and $m = 2^n-1$. By definition, $p_1 \equiv p$. It is important to know something about the values of p_s in order to use the block substitutions which they generate. It is obvious that $p_s \neq p_t$ if $s \neq t$.

Proposition 8: $p \geq n$

Proof: The (p+1)st row of the basic array is:

$$x_p \oplus x_{p+1} = x_{p+1-p} = x_1$$

If $p+1 \leq n$, this implies that the first n rows are dependent which contradicts proposition 7. (Q.E.D.)

Note: Since s is a power of a permutation g of order m, if $s = m+k$

$$g^{m+k} = g^m \circ g^k = I \circ g^k.$$

So the exponents can be expressed modulo m.

Proposition 9: If p is an allowable value for the shift, so is $\overline{p} = 2^n-p$.

Proof: In the normalized array, the first column is displaced down one place with respect to column 2. The third column is displaced down p places (the shift) with respect to column 2 and p-1 places from column 1. If we interchange columns 1 and 2 and turn the resulting array upside down (i.e., reverse the order of the rows) then in the new array, the third column is shifted down $m-(p-1) = 2^n-p$ places with respect to column 2. Since only the order has been changed, the new array is equivalent to the original in the sense that it will generate permutations from the same group. (Q.E.D.)

Note: Since $\overline{p} \geq n$, $p \leq 2^n-n$.

Proposition 10: If the power $s = 2^r$ for some integer $r \geq 0$, $p_s = sp$.

Proof: If $r = 1$, $s = 2$ and $x_{k-2} \oplus x_k = x_{k-p_2}$. One can write:

$$x_{k-2} \oplus x_k = x_{k-2} \oplus x_{k-1} \oplus x_{k-1} \oplus x_k.$$

Considering k-1 and k-p as indices,

$$x_{k-2} \oplus x_k = x_{k-1-p} \oplus x_{k-p} = x_{k-p-p} = x_{k-2p}.$$

So $p_2 = 2p$ and the statement is true for $r = 1$.

Assume that for some $s = 2^r$, $p_s = 2^rp$. $2^{r+1} = 2^r + 2^r$.
Then,

$$x_{k-2^{r+1}} \oplus x_k = x_{k-2^r-2^r} \oplus x_k = x_{k-2^r-2^r} \oplus x_{k-2^r} \oplus x_{k-2^r} \oplus x_k$$
$$= x_{k-2^r-2^rp} \oplus x_{k-2^rp} = x_{k-2^rp-2^rp} = x_{k-2^{r+1}p}. \qquad \text{(Q.E.D.)}$$

There are other constraints on p, the shift in the third column. If g is the permutation defined by columns 2 ⟶ 1 of the additive array, then the permutation defined by columns 1 ⟶ 3 is g^{p-1} and by columns 3 ⟶ 2, is g^{m-p}. These permutations are obviously commutative and $g \circ g^{p-1} \circ g^{m-p} = g^0 = I$, the identity permutation.

The sum of the exponents $1 + (p-1) + (m-p) \equiv 0 \bmod m$. Clearly, $1 < p < m$. More generally, for a power g^s, the corresponding permutations are:

$$g^s \circ g^{p_s-s} \circ g^{m-p_s} = I$$

Then $s + (p_s-s) + (m-p_s) \equiv 0 \bmod m$ and $p_s \neq s, m$.

Proposition 11: If an additive array generates the power s of the basic permutation in two pairs of columns, it generates this in all three columns and $3s \equiv 0 \bmod m$.

Proof: If two of the three permutations are of the same power, the array can be arranged so that typical rows are:

$$\begin{array}{c}
x_{k-s} \oplus x_k = x_{k+s} \\
\cdot \\
\cdot \\
\cdot \\
x_k \oplus x_{k+s} = x_{k+2s}
\end{array}$$

By the group property, adding these together will give another row in the same array:

$$\begin{aligned}
x_{k+s} \oplus x_{k+2s} &= (x_k \oplus x_{k+s}) \oplus (x_k \oplus x_{k-s}) \\
&= x_{k+s} \oplus x_{k-s}
\end{aligned}$$

which implies that $x_{k+2s} = x_{k-s}$ and that $3s \equiv 0 \bmod m$. This will occur only if $3 \mid m$, in which case, $p_s = -s \equiv 2s \bmod m$, and $p_s - s = 2s - s = s$, $m - p_s = s$.

(Q.E.D.)

Corollary: If an additive array generates the same permutation with each pair of columns, the permutation is of order 3.

Proof: If $h = g^s$ is generated as in proposition 11, then $h^3 = g^{3s} = I$.

(Q.E.D.)

If $p_s = 2s$, $\frac{m+s}{2}$ or $m-s$, the same permutation will be generated, so if $3 \nmid m$, $p_s \neq \{ s, 2s, \frac{m+s}{2}, m-s, m \}$.

There are a number of miscellaneous facts to help select or eliminate candidates for the shift, but so far insufficient rules have been found to fully determine the shift pattern. Some of the facts are:

1. If $p_s = t$, then $p_t = s$ by symmetry

2. If $p = 2^t$, then

   $$x_{k-1} = x_k \oplus x_{k-p} = x_k \oplus x_{k-2^t} = x_{k-2^t p} = x_{k-2^{2t}}$$

   which is possible only if $2^{2t} \equiv 1$ modulo m.

3. If $p = 2^t+1$, then

   $$x_k = x_{k-1} \oplus x_{k-1-2^t} = x_{k-1-2^t p} = x_{k-1-2^t(2^t+1)}$$

   which is possible only if $2^{2t} + 2^t + 1 \equiv 0$ modulo m.

4. If $p \mid m$ then $\frac{m}{p} \neq 2^t - 1$ since otherwise, $2^t p = m+p$ and

   $$x_k \oplus x_{k-2^t} = x_{k-2^t p} = x_{k-m-p} = x_{k-p} = x_k \oplus x_{k-1}$$ and $t = 0$.

ENUMERATION OF MAXIMAL ADDITIVE PERMUTATIONS

The additive arrays, by convention, have been constructed according to the permutation defined by columns 1 and 2. If the permutation is of maximal length (i.e., a permutation of $m = 2^n-1$ elements with no subcycles) the array will be maximal also in the sense that it has no distinct self-contained blocks. Conversely, the maximal array will generate a maximal permutation. However, powers of a maximal permutation may have cycles. This will be the case unless the maximal length permutation is of prime order. Thus, no general conclusions can be drawn about the permutations generated by other pairs of columns.

If all six of the permutations generated by the additive array have the same cycles, the array will have corresponding cycles in the form of disjoint subarrays of the same length in which each column has the same elements. If the basic permutation generated by columns 1 and 2 has a cyclical structure not shared by the other permutations, then there will be disjoint subarrays or blocks in the array in which the elements in column 3 will be distinct from those in columns 1 and 2.

It is also obvious that any maximal length additive permutation can be used to generate a maximal array. Thus, initially considering arrays and permutations with fixed identity, there is a one-to-one correspondence between maximal length permutations and columns 1 and 2 of maximal arrays and, hence, with the array as an entity.

Proposition 12: For n-bit binary numbers, the number of maximal length additive arrays with fixed identity for a given value of the shift p, is

$$M(n) = \prod_{k = 1}^{n-1} (2^n-2^k) \quad \text{where } m = 2^n-1.$$

Proof: Let Q(n) be the number of distinct sets of generators of the n-bit binary numbers. Each set of generators will generate a maximal array. However, the order in which the generators are used makes a difference. This is because the column 3 lead term $x_m = f\ (x_1, \ldots x_n)$ depends on the relative

orders of the generators. There are n! such orders. Once an array is generated, the starting row of the array or starting point of the permutation is irrelevant. For example, if the first row is: $x_m \oplus x_1 = x_{1-p}$ it could be circulated to the bottom and instead start with: $x_1 \oplus x_2 = x_{2-p}$ so the process is redundant by a factor of $m = 2^n-1$.
Therefore:

$$M(n) = \frac{n!}{m} Q(n)$$

Assume $Q(n,k) = \frac{1}{k!} \prod_{j=0}^{k-1} (2^n-2^j)$ is the number of independent k-tuples which can be selected from n-bit words. Clearly, it is true for k = 1 or k = 2. n-tuples can be constructed by adjoining a single n-bit word to an (n-1)-tuple. There are m such words which could be adjoined. However, $m = 2^n-1$ must be diminished by eliminating the $n-1 = \binom{n-1}{1}$ words which duplicate those in the (n-1)-tuple, by the $\binom{n-2}{2}$ words which are sums of pairs of words in the (n-1)-tuple, etc, and finally by the $1 = \binom{n-1}{n-1}$ word which is the sum of all words in the (n-1)-tuple. Thus, m must be diminished by $\sum_{j=1}^{n-1} \binom{n-1}{j} = 2^{n-1}-1$.

The number of words to be adjoined to the (n-1)-tuples $= 2^n-1-(2^{n-1}-1) = 2^n-2^{n-1}$. However, each n-tuple could be generated by n different (n-1)-tuples, so that,

$$Q(n) = Q(n,n) = \frac{2^n-2^{n-1}}{n} Q(n,n-1) = \frac{2^n-2^{n-1}}{n} \frac{1}{(n-1)!} \prod_{k=0}^{n-2} (2^n-2^k) = \frac{1}{n!} \prod_{k=0}^{n-1} (2^n-2^k)$$

Thus, $$M(n) = \frac{1}{m} \prod_{k=0}^{n-1} (2^n-2^k) = \prod_{k=1}^{n-1} (2^n-2^k)$$ (Q.E.D.)

M(n) can be rewritten in a form easier to compute:

$$M(n) = \prod_{k=1}^{n-1} (2^n-2^k) = 2^s \prod_{j=1}^{n-1} (2^j-1)$$

where $s = \frac{n(n-1)}{2}$. There is a simple recursive relationship:

$$M(n+1) = 2^n\ (2^n-1)\ M(n)$$

Unless m is prime, the permutation group will contain some permutations which are not maximal and, thus, will have some proper cycles. These will be powers g^s of a maximal permutation g where s is a divisor of m. These can be enumerated by counting the number of values of s which are divisors of m-1.

In a group of additive permutations of n-bit numbers with fixed I, there are L(n) maximal permutations and S(n) non-maximal permutations.

$$L(n) + S(n) = m-1 = 2^n-2$$

While there are $2^n!$ permutations on the set of n-bit numbers, there are only $2^{n-1}!$ which give distinct substitutions because the starting point in the permutation does not make any difference. In the maximal additive array, we can cycle the order of the rows, other than the fixed row, and not change the substitution. These additive arrays generate permutations which are a subset of all possible permutations having a single fixed point. If F(n) is the total number of permutations holding any one of the 2^n words fixed, then

$$F(n) = 2^n(2^n-2)! = \frac{2^n!}{m}.$$

For convenience, a number of other terms can be defined. They are as follows:

$N_p(n)$ - the number of distinct values of p_1, the basic shift in column 3 of a maximal additive array of n-bit numbers.

M(n) - the number of maximal additive permutations for a given p and a given fixed point. It is derived in proposition 12.

T(n) - the total number of additive permutations for a given p and a given fixed point. If $m = 2^n-1$ is prime, then T(n) = M(n). In general:

$$T(n) = \frac{m-1}{L(n)} M(n).$$

H(n) - the total number of additive permutations on n-bit numbers:

$$H(n) = 2^n N_p(n) T(n).$$

G(n) - the number of groups of additive permutations on n-bit numbers:

$$G(n) = \frac{H(n)}{m-1} = \frac{2^n N_p(n) M(n)}{L(n)}.$$

Table 1 is a tabulation of these parameters for bit size $n \leq 8$. Ratio $\frac{H(n)}{F(n)}$ gives a measure of the relative rarity of additive permutations.

UNBIASED BLOCK SUBSTITUTIONS

In designing a block substitution device, a basic goal is to have an unbiased transformation of the n-bit binary numbers onto themselves. This does not seem to be well defined but, basically, one desires that the cipher text contain no information on the nature of the clear text. For example, no subset should be mapped onto itself, and numbers with some characteristic (such as, even, lower half, zero in a binary position) should be equally likely to be images of a number with the same property or the reverse property. To require

Table 1. Comparison of Permutations of Various Bit Sizes

n	3	4	5	6	7	8
2^n	8	16	32	64	128	256
m	7	15	31	63	127	255
$N_p(n)$	2	2	6	6	18	16
$M(n)$	24	1,344	322,560	3.20×10^{8}	1.29×10^{12}	8.42×10^{16}
$L(n)$	6	8	30	36	126	128
$S(n)$	0	6	0	26	0	126
$T(n)$	24	2,352	322,560	5.51×10^{8}	1.29×10^{12}	16.78×10^{16}
$H(n)$	384	75,264	61,931,520	2.12×10^{11}	2.97×10^{15}	6.87×10^{20}
$G(n)$	64	5,376	2,064,384	3.41×10^{9}	2.36×10^{13}	2.71×10^{18}
$F(n)$	5760	1.39×10^{12}	8.49×10^{33}	2.01×10^{87}	3.04×10^{213}	3.36×10^{504}
$\frac{H(n)}{F(n)}$	.0667	5.40×10^{-8}	7.30×10^{-27}	1.05×10^{-76}	9.79×10^{-199}	2.04×10^{-484}

that all outputs be equally likely for a random input is insufficient and requires only that the transformation be one-to-one. The following definition is proposed.

Definition: A transformation or substitution of the n-bit binary numbers onto themselves is said to be unbiased if:

a. There is no invariant subset other than a single fixed point.

b. Every maximal subgroup of the n-bit numbers is mapped in two equal parts into itself and into its relative complement.

Before proceding further, it is necessary to collect some facts on these maximal subgroups. Any set of binary numbers that has zeros in the same binary position is such a subgroup; however, there are others as well. Any subgroup of order 2^k is embedded in a subgroup of order 2^{k+1} and has a relative complement in the larger subgroup with 2^k elements.

Proposition 13: Let G_n be the group of n-bit binary numbers with modulo 2 addition as the group operation. Let H be a subgroup of order 2^{n-1} and $\overline{H}$ its complement in G_n. Then for $x, y \in \overline{H}$, $x \oplus y \in H$, and if $x \in H$, and $y \in \overline{H}$, $x \oplus y = z \in \overline{H}$.

Proof: $\overline{H}$ cannot be a subgroup since it does not contain the identity $I = (0\cdots0)$. By definition, if $x, y \in H$, $x \oplus y \in H$.

Let $z = x \oplus y$ where $x \in H$ and $y \in \overline{H}$. If $z \in H$, then $y = z \oplus x \in H$ by the group property. This is a contradiction since $H \cap \overline{H} = \emptyset$. So $z \in \overline{H}$. The number of elements in H and in $\overline{H}$ is $p = 2^{n-1}$. Each element or binary number in H can be written as the sum of p pairs of numbers belonging to H. This gives p^2 pairs from H to H.

The numbers in $\overline{H}$ which are sums of mixed pairs can be expressed in two ways considering order: p^2 pairs in which the p numbers from H are matched with the p numbers from $\overline{H}$. Another p^2 pair is generated by matching the p words from $\overline{H}$ with p words from $\overline{H}$. This gives a total of $2p^2$ ways of expressing the mixed sums in $\overline{H}$.

Overall, in G there are 2p words which can be expressed as $(2p)^2 = 4p^2$ pairs. The remaining p^2 pairs are those where the sum of numbers from $\overline{H}$ is taken. So far, $2p^2$ pairs correspond to $\overline{H}$ and p^2 pairs to H. Since the two sets H and $\overline{H}$ have the same number of elements, the remaining p^2 pairs belong to H (i.e., $x, y \in \overline{H} \Rightarrow x \oplus y \in H$). (Q.E.D.)

A substitution of the n-bit binary numbers G_n onto themselves can also be thought of as a transformation T, such that, $TG_n = G_n$ if this is expressed by an additive array:

$$\begin{array}{c} y_1 \oplus x_1 = z_1 \\ \cdot \\ \cdot \\ \cdot \\ y_k \oplus x_k = z_k \\ \cdot \\ \cdot \\ \cdot \end{array}$$

Then the first column defines the transformation $Tx_k = z_k$. If T is linear, $T(x_j \oplus x_k) = Tx_j \oplus Tx_k$.

<u>Proposition 14:</u> A maximal additive array with fixed identity defines a linear transformation.

<u>Proof:</u> A general row is $x_{k-1} \oplus x_k = x_{k-p}$.
Let T be the transformation mapping column 2 on column 3, $Tx_k = x_{k-p}$.
Let $x_k \oplus x_j = x_i$.
$T(x_k \oplus x_j) = Tx_i = x_{i-p}$.
$Tx_k \oplus Tx_j = x_{k-p} \oplus x_{j-p}$.
Since the array is an additive group by proposition 3,

$$x_{k-p} \oplus x_{j-p} = (x_k \oplus x_j) \oplus (x_{k-1} \oplus x_{j-1}) = x_i \oplus x_{i-1} = x_{i-p}$$

and

$$T(x_k \oplus x_j) = Tx_k \oplus Tx_j.$$

This property is not generally true for one-to-one transformations on G_n.

Proposition 15: A transformation of the n-bit binary numbers onto themselves can be linear only if I = (00···0) is a fixed point.

Proof: Without loss of generality, the n-bit binary numbers can be arranged in order such that the successor to each is its image under the transformation:

$$\begin{array}{c} x_1 \longrightarrow x_2 \\ x_2 \longrightarrow x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_k \longrightarrow x_{k+1} \\ \cdot \\ \cdot \\ \cdot \\ x_m \longrightarrow x_1 \end{array}$$

Since T is linear, $T(x_k \oplus x_j) = Tx_k \oplus Tx_j = x_{k+1} \oplus x_{j+1}$. $x_k \oplus x_j = x_i$ for some i in the set 1,....,m. If $x_j = I$, then $T(x_k \oplus x_j) = T(x_k \oplus I) = Tx_k = Tx_k \oplus Tx_j$, which implies that $Tx_j = x_{j+1} = I$. (Q.E.D.)

Proposition 16: A linear transformation of the n-bit binary numbers onto themselves can be represented as an additive array.

Proof: Without loss of generality, the n-bit numbers can be arranged in an order so that each is mapped onto its successor:

$$\begin{array}{c} x_1 \longrightarrow x_2 \\ x_2 \longrightarrow x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_k \longrightarrow x_{k+1} \\ \cdot \\ \cdot \\ \cdot \\ x_m \longrightarrow x_1 \end{array}$$

By proposition 15, if T is linear, I = (0···0) is a fixed point. Initially, it is assumed that there are no cycles. $Tx_k = x_{k+1}$. Since T is a linear transformation, $T(x_k \oplus x_j) = Tx_k \oplus Tx_j$. If $x_i = x_k \oplus x_j$, then

$$x_{i+1} = Tx_i = T(x_k \oplus x_j) = Tx_k \oplus Tx_j = x_{k+1} \oplus x_{j+1}.$$

Let $z_k = x_k \oplus x_{k-1}$ for each of the m = 2^{n-1} binary numbers other than I = (0···0). There are m = 2^{n-1} such equations. Either z_k takes on all values of the n-bit numbers except I, or there are duplicates. Suppose $z_k = z_j$ for some $k \neq j$. Then $x_k \oplus x_{k+1} = x_j \oplus x_{j+1}$ and $x_i = x_k \oplus x_j = x_{k+1} \oplus x_{j+1} = x_{i+1}$, a contradiction.

If there are cycles or invariant sets, each cycle can be written as above as a self-contained set and the same reasoning applied to each cycle. (Q.E.D.)

So far we have seen that all linear transformations on G_n are additive with fixed identity. Next we consider affine transformations which are non-linear (at least to a mathematician). An affine transformation has the property that $T(x \oplus y) = Tx \oplus Ty \oplus C$ where C is some constant.

Proposition 17: A maximal additive array with a fixed point other than the identity defines an affine transformation.

Proof: It has already been shown that an additive array with some fixed number A can be obtained from an array with fixed I by adding the row $A \oplus A = I$ to each row in that array.

In the array with fixed I, $Tx_k = x_{k-p}$. In the array with fixed A, $T'(x_k \oplus A) = x_{k-p}$. Then, redesignating

$$\dot{x}_j = x_k \oplus A,\ T'x_j = Tx_k = T(x_j \oplus A).$$
$$T'(x \oplus y) = T(x \oplus y \oplus A) = T(x \oplus A \oplus y \oplus A \oplus A)$$
$$T'(x \oplus y) = T(x \oplus A) \oplus T(y + A) + TA$$
$$T'(x \oplus y) = T'x \oplus T'y \oplus C \text{ where } C = TA. \qquad \text{(Q.E.D.)}$$

If T' is an arbitrary affine transformation, it can be written in terms of a linear transformation T as $T'x = Tx \oplus C$. By proposition 16, T can be represented as an additive array. Defining $A = T^{-1}C$, a corresponding additive array can be generated to represent T' by adding the row $A \oplus A = I$ to each row in the array for T.

Proposition 18: If H is a subgroup of G_n, the group of n-bit binary numbers, and T is a linear transformation or $T'x = Tx \oplus TA$ with $A \in H$, is an affine transformation on G_n, then TH and T'H are subgroups of G_n.

Proof: Let G = TH. By propositions 15 and 16, T can be represented by an additive array with fixed identity. $I \in H$, $TI = I$ so $I \in G$. Let $x, y \in H$. Since H is an additive group $x \oplus y = z \in H$. Tx, Ty and $Tz \in G$. By linearity $Tz = T(x \oplus y) = Tx \oplus Ty \in G$, so G is closed under modulo 2 addition. For any $x \in G$, $x \oplus x = I \in G$, so x is its own inverse and G is a group. In the affine case, $TA \in G$ and $T'A = TA \oplus TA = I$, so $I \in G$. $T'x \oplus T'y = T'(x \oplus y \oplus A) \in G$ since $x \oplus y \oplus A \in H$. (Q.E.D.)

Proposition 19: A maximal length linear or affine transformation is unbiased.

Proof: First of all, being of maximal length, the transformation has no cycles, other than its fixed point, and thus has no proper invariant set.

Let T be linear and consider a maximal subgroup H of G_n. Define two sets:

$$M = \{x \in H \mid Tx \in H\}$$
$$N = \{x \in H \mid Tx \in \overline{H}\}$$
$$I \in M,\ M \cap N = \emptyset \text{ and } H = M \cup N.$$

If $x, y \in M$, $T(x \oplus y) = Tx \oplus Ty \in H$ since T is linear and H is a group. Thus, $x \oplus y \in M$ also. Since each x is a self-inverse, M is a subgroup, and N is its complement relative to H.

If $x \in M$ and $y \in N$, then $z = x \oplus y \in H$ and, hence, is either in M or N. If $z \in M$, then $y = z \oplus x \in M$ by its group property. Thus, $z \in N$.

If $x, y \in N$, $Tx, Ty \in \overline{H}$, complement of a maximal subgroup H. By proposition 13, $Tx \oplus Ty \in H$. By linearity $T(x \oplus y) \in H$. Since $x, y \in H$, $x \oplus y \in H$. Since $T(x \oplus y) \in H$, so $x, y \in N \Longrightarrow x \oplus y \in M$.

Since T has no proper invariant subset, $TH \neq H$ and $N \neq \emptyset$, the empty set. Select $x \in N$ and consider the group Q generated by x and M. $Q \subset H$. Q contains the complex $x \oplus M$ which includes all pairs of the form $x \oplus y$ where $y \in M$. Obviously, $M \subset Q$. Choose any $z \in N$, where $z \neq x$. This is possible since $\left| M \right|$ divides $\left| H \right| = 2^{n-1}$. So $\left| N \right| \geq 2$. $z = x \oplus w$ for some w. Since $z, x \in H$, a group, $w \in H$ also. Since $x, z \in N$, $w \in M$. Thus, any element of N or M is an element of Q. Therefore, $Q = H$ and M is a maximal subgroup of H. H is of order 2^{n-1} and, thus, the order of the maximal subgroup M is $\left| M \right| = 2^{n-2} = \frac{1}{2} \left| H \right|$. $\left| N \right| = \left| H \right| - \left| M \right| = \frac{1}{2} \left| H \right|$.

Note: TM is a group and $\left| TM \right| = \left| M \right|$.

Let T' be an affine transformation defined by $T'x = Tx \oplus TA$, and $C = TA$. Again define two sets:

$$M' = \{ x \in H \mid T'x \in H \}$$
$$N' = \{ x \in H \mid T'x \in \overline{H} \}$$

Assume $C \in H$. If $x \in M'$, $T'x = Tx \oplus C \in H$. Thus $Tx \in H$ and $M' \subseteq M$. If $x \in M$, $Tx \in H$. $T'x = Tx \oplus C \in H$, so $x \in M'$, and $M' = M$. Since $M' \cup N' = H$ and $M' \cap N' = \emptyset$, $N' = N$. From here, the proof proceeds as in the first part.

Now assume $C \in \overline{H}$. If $x \in M'$, $T'x = Tx \oplus C \in H$. Thus, $Tx \in \overline{H}$ and $M' \subseteq N$. If $x \in N$, $Tx \in \overline{H}$. $T'x = Tx \oplus C \in H$, $M' = N$ and, consequently, $N' = M$. In this case, M' is not a subgroup but, by the first part of the proof, $\left| M' \right| = \left| N \right| = \frac{1}{2} \left| H \right|$. Since H is maximal, the relative complement $\overline{H}$ is the total complement, so that $C \in H$ or $C \in \overline{H}$. (Q.E.D.)

If T is linear, the proof holds for any subgroup. If H is of order 2^{n-1} it is isomorphic to the group G_{n-1} of n-1 bit binary numbers. G_{n-1} has a relative complement in G_n consisting of $\bar{H}$. The same process can be applied to G_{n-1}. This leads to a nested sequence of subgroups

$$G_n \supset G_{n-1} \supset \cdots \; G_1 \supset G_0 \;\; = \{ I \}$$

in which each G_k is mapped by T half onto G_{k-1} and half onto $\overline{G_{k-1}}$.

Note: This sequence of subgroups is not unique, as we can begin with several candidates for G_{n-1}, etc.

We have seen that any additive array produces unbiased substitutions or transformations. Next, we consider the necessary properties for a transformation or substitution to be unbiased in the sense defined here.

<u>Proposition 20:</u> G_n has $m = 2^n - 1$ maximal subgroups.

<u>Proof:</u> G_n is generated by any n independent n-bit numbers. Let $\{ x_1, x_2, \cdots, x_n \}$ be such a set. Any subset of n-1 of the generators will generate a maximal subset. $\binom{n}{n-1} =$ n maximal subgroups will be generated this way. Let this collection of maximal subgroups be designated $H_1, H_2, \cdots, H_n$ where H_i is generated by $\{ x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n \}$, that is, by all except x_i. Then $x_i \in H_i$ for each $i \leq n$, and $x_i \in H_j$ for $i \neq j$.

Next, one can define an operation "+" between maximal subgroups as:

$$H_k = H_i + H_j \equiv (H_i \cap H_j) \cup (\bar{H}_i \cap \bar{H}_j).$$

$\left| H_k \right| = 2^{n-1}$ since it is a union of disjoint sets and $\left| H_i \cap H_j \right| = \left| \bar{H}_i \cap \bar{H}_j \right| = 2^{n-2}$. Let $x_c = x_a \oplus x_b$ where $x_a, x_b \in H_k$. There are three possibilities:

1. $X_a, x_b \in H_i \cap H_j$, then $x_c \in H_i \cap H_j$ since $H_i \cap H_j$ is a group.

2. $x_a, x_b \in \overline{H_i} \cap \overline{H_j}$, then $x_c \in H_i \cap H_j$ by proposition 13.

3. $x_a \in H_i \cap H_j$ and $x_b \in \overline{H_i} \cap \overline{H_j}$, then $x_c \in \overline{H_i} \cap \overline{H_j}$ by proposition 13.

In each case $x_c \in H_k$, obviously $I \in H_k$, so H_k is also a maximal subgroup.

To each maximal subgroup generated in this fashion, we can associate $x_k = x_i \oplus x_j$, that is, x_k is the sum of generators in the set $\{ x_1, x_2, \cdots , x_n \}$ which belong to $\overline{H_k}$. Clearly, the maximal subgroups generated this way total m and are in one-to-one correspondence to the n-bit numbers.

Assume there is some maximal subgroup H_0 which was not generated this way. Let the q numbers $\{ x_a, x_b, \cdots \} \in \overline{H_0}$ be the maximal such subset of the generators $\{ x_1, x_2, \dots , x_n \}$. $x_0 = x_a \oplus x_b \oplus \cdots$. $x_0 = x_k$ for some H_k previously generated. Since the generators produce unique sums, then $\{ x_a, x_b, \cdots \} \in \overline{H_k}$. There are p remaining generators $\{ x_f, x_g \cdots \} \in \overline{H_0} \cap \overline{H_k}$ where $p + q = n$. Then $(H_0 \cap H_k) \cup (\overline{H_0} \cap \overline{H_k})$ is a maximal subgroup. This is not possible unless $H_0 = H_k$. (Q.E.D.)

There is an interesting relationship between the maximal subgroups and the numbers. G_n is trivially a maximal subgroup of itself. $G_n + H_i = H_i + G_n = H_i$ and $H_i + H_i = G_n$. The collection of maximal subgroups $\hat{G} = \{ G_n, H_i, \cdots , H_n, H_1 + H_2, \cdots \}$ form a group under the operation "+" with G_n acting as the identity. $\hat{G}$ is isomorphic to G_n under the mapping $x_k \longrightarrow H_k$ defined above.

The above proof is a little complicated, but it illustrates something of the structure of the subgroups. The same result can be obtained by more elementary means.

Each set of n-1 independent elements generates a maximal subgroup of order 2^{n-1}. The number of distinct sets of independent (n-1)-tuples, from proposition 12, is Q (n, n-1). However, different (n-1)-tuples may generate the same subgroup. The subgroup of order n-1 is isomorphic to the group of n-1 bit numbers. This latter group has Q(n-1) independent (n-1)-tuples. Thus, the group of n-1 bit numbers can be generated by Q(n-1) sets of generators, and each maximal subgroup is generated Q(n-1) times using all possible independent (n-1)-tuples. So the number of distinct maximal subgroups of G_n is:

$$R(n) = \frac{Q(n,\ n-1)}{Q(n-1)} = \frac{\dfrac{1}{(n-1)!} \displaystyle\prod_{j=0}^{n-2} (2^n - 2^j)}{\dfrac{1}{(n-1)!} \displaystyle\prod_{j=0}^{n-2} (2^{n-1} - 2^j)} =$$

$$\prod_{j=0}^{n-2} \left(\frac{2^n - 2^j}{2^{n-1} - 2^j} \right) = 2^n - 1 = m$$

Corollary: Each number other than the identity will occur in $\frac{m - 1}{2}$ maximal subgroups and in $\frac{m + 1}{2}$ complements.

Proof: Since there are m maximal subgroups each number will appear m times, either in a maximal subgroup or its complement. The identity I will appear m times in the maximal subgroups but never in a complement. There are $\frac{m\ (m-1)}{2}$ remaining places for the non-identity numbers to appear in maximal subgroups and $\frac{m\ (m+1)}{2}$ places in the complements. Thus, by symmetry, each non-identity number will appear in $\frac{m-1}{2}$ maximal subgroups and in $\frac{m+1}{2}$ complements. (Q.E.D.)

Proposition 21: If H_i, H_j, H_k are any three dependent maximal subgroups, then $G_n = H_i + H_j + H_k$.

Proof: Each H_i and each $\overline{H_i}$ have $\frac{m+1}{2}$ numbers. Also each $H_i \cap H_j$, $H_i \cap \overline{H_j}$, and $\overline{H_i} \cap \overline{H_j}$ have $\frac{m+1}{4}$ numbers. The set of distinct numbers in $H_i \cup H_j$ is the set $(H_i \cap \overline{H_j}) \cup (\overline{H_i} \cap H_j) \cup (H_i \cap H_j)$ and consists of $\frac{3(m+1)}{4}$ numbers. The missing $\frac{m+1}{4}$ numbers are those contained in $\overline{H_i} \cap \overline{H_j}$. $H_k = H_i + H_j = (H_i \cap H_j) \cup (\overline{H_i} \cap \overline{H_j})$. So $G_n = H_i \cup H_j \cup H_k$ and $G_n = H_i + H_j + H_k$. (Q.E.D)

Corollary: The set of any three dependent numbers $\{ x_i, x_j, x_k \}$, none of which is I, contain at least one number from each of the m maximal subgroups.

Proof: In the proof of proposition 20, for a set of generators of G_n, $\{ x_1, x_2, \cdots, x_n \}$, a set of maximal subgroups was generated. The generators can be written in terms of these subgroups:

$$\begin{aligned} \{ x_1 \} &= \overline{H_1} \cap H_2 \cap \cdots\cdots \cap H_n \\ \{ x_2 \} &= H_1 \cap \overline{H_2} \cap \cdots\cdots \cap H_n \\ &\;\;\vdots \\ \{ x_n \} &= H_1 \cap H_2 \cap \cdots\cdots \cap \overline{H_n} \end{aligned}$$

$|H_k| = |\overline{H_k}| = 2^{n-1}$, $|H_j \cap H_k| = |\overline{H_j} \cap \overline{H_k}| = |H_j \cap \overline{H_k}| = |\overline{H_j} \cap H_k| = 2^{n-2}$. By induction these n-fold intersections have $2^0 = 1$ elements and are singleton sets. As an example, take x_1, x_2, and $x_k = x_1 \oplus x_2$. $H_k = H_1 + H_2$ is a dependent set in the sense of proposition 20, x_k is the number associated with H_k, and by proposition 21, $G_n = H_1 + H_2 + H_k$. By proposition 13,

$$\{x_k\} = \overline{H_1} \cap \overline{H_2} \cap H_3 \cap \cdots \cap H_n$$

of course, $k \notin \{1, 2, \cdots, n\}$. Any maximal subgroup is of the form $\sum H_i$ where i ranges over some subset of $\{ 1, 2, \cdots, n \}$. If $i \neq 1, 2$, then $\{ x_1, x_2, x_k \} \in \sum H_i$. Otherwise, there are three possible cases:

1. $i \neq 1, = 2 \qquad x_1 \in \sum H_i$

2. $i = 1, \neq 2 \qquad x_2 \in \sum H_i$

3. $i = 1, = 2 \qquad x_k \in \sum H_i$ (Q.E.D.)

This is different from the situation with vector spaces where an independent set of vectors spans the space.

<u>Proposition 22:</u> A one-to-one transformation of G_n onto itself is unbiased, if and only if, it can be represented by a replicative array.

<u>Proof:</u> As shown in the introduction, any transformation $TG_n \rightarrow G_n$, where $Tx_i = Z_i$, can be written in the form $y_i \oplus x_i = z_i$. This can be represented by the addition modulo 2 of two column matrices to obtain a third:

$$\begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} \oplus \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}$$

The y_i matrix represents the transformation T. Each matrix will have 2^n entries. For the x_i and z_i matrices, each of the 2^n n-bit binary numbers will appear exactly once since T is 1 to 1 onto it. Assume that T is unbiased and that the y_i matrix may have some of the n-bit numbers repeated so that the 2^n numbers are not all distinct.

There are $m = 2^n-1$ maximal subgroups $H_1, H_2, \cdots, H_m$. Since T is unbiased, it will map each H_i such that

$$\left| TH_i \cap H_i \right| = \left| TH_i \cap \overline{H_i} \right| = \frac{1}{2} \left| H_i \right| .$$

From proposition 13, we can arrange the matrices in blocks that correspond to the maximal subgroup H_i and its complement

$$\begin{array}{ccccc} y & & x & & z \\ \left(\begin{array}{c} H_i \\ \hline \overline{H}_i \\ \hline \overline{H}_i \\ \hline H_i \end{array}\right) & \oplus & \left(\begin{array}{c} H_i \\ \hline H_i \\ \hline \overline{H}_i \\ \hline \overline{H}_i \end{array}\right) & = & \left(\begin{array}{c} H_i \\ \hline \overline{H}_i \\ \hline H_i \\ \hline \overline{H}_i \end{array}\right) \end{array}$$

Since T is unbiased, each of the four blocks must consist of 2^{n-2} numbers. The n-bit numbers may be designated $y_0 = I$, y_1, y_2, $\cdots$, y_n in the y matrix, and each will appear with some multiplicity $p_j \geq 0$. Let $a_{ij} = 1$ if $y_j \in H_i$ and $a_{ij} = 0$ if $y_j \in \overline{H}_i$. Clearly $a_{i0} = 1$ for all i since the identity I belongs to each subgroup. For each H_i and $\overline{H}_i$ the following equations hold:

$$H_i : \quad p_0 + \sum_{j=1}^{m} a_{ij} p_j = \frac{m+1}{2}$$

$$\overline{H}_i : \quad \sum_{j=1}^{m} (1 - a_{ij}) p_j = \frac{m+1}{2}$$

These two sets of equations are consistent since

$$p_0 + \sum_{j=1}^{m} p_j = \sum_{j=0}^{m} p_j = m + 1.$$

The m equations corresponding to the m maximal subgroups can be written in matrix form:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{pmatrix} = \begin{pmatrix} \frac{m+1}{2} - p_o \\ \frac{m+1}{2} - p_o \\ \vdots \\ \frac{m+1}{2} - p_o \end{pmatrix}$$

Using the determinant of the m x m matrix on the left, we can use Cramer's rule to solve for the p_i. For example,

$$p_1 = \frac{\begin{vmatrix} \frac{m+1}{2} - p_o & a_{12} & a_{13} & \cdots & a_{1m} \\ \frac{m+1}{2} - p_o & a_{22} & a_{23} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{m+1}{2} - p_o & a_{m2} & a_{m3} & \cdots & a_{mm} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mm} \end{vmatrix}}$$

From the corollary to proposition 20, each row of the determinant in the denominator will have $\frac{m-1}{2}$ entries which are 1 and $\frac{m+1}{2}$ which are 0. Adding columns 2 through m to column 1 will not change the value of the determinant, so that:

$$p_1 = \frac{\begin{vmatrix} \frac{m+1}{2} - p_o & a_{12} & a_{13} & \cdots & a_{1m} \\ \frac{m+1}{2} - p_o & a_{22} & a_{23} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{m+1}{2} - p_o & a_{m2} & a_{m3} & \cdots & a_{mm} \end{vmatrix}}{\begin{vmatrix} \frac{m-1}{2} & a_{12} & a_{13} & \cdots & a_{1m} \\ \frac{m-1}{2} & a_{22} & a_{23} & \cdots & a_{2m} \\ \vdots & & & & \vdots \\ \frac{m-1}{2} & a_{m2} & a_{m3} & \cdots & a_{mm} \end{vmatrix}}$$

Factoring out the constant first column in each determinant and noting that the same result holds for the ith column, yields:

$$p_i = \frac{\frac{m+1}{2} - p_o}{\frac{m-1}{2}} \qquad \text{for } 1 \leq i \leq m.$$

$p_0 \leq 1$ since $p_0 > 1$ implies more than one fixed word, contrary to the definition of an unbiased substitution. If $p_0 = o$ then $P_i = \frac{m+1}{m-1}$ which is not an integer.

The only remaining posibility is $p_0 = 1$, in which case:

$$p_i = \frac{\frac{m+1}{2} - 1}{\frac{m-1}{2}} = 1 \text{ for all } i.$$

Thus if the transformation is unbiased, all the $p_i = 1$ which, by definition, is a replicative array.

NOTE: One does not need to appeal to the definition of an unbiased substitution to restrict the value of p_0. If $p_0 \geq 2$, than $o < p_i < 1$ which is also impossible.

Assume now that the transformation is replicative. Consider any arbitrary maximal subgroup H. Each column matrix will have 2^{n-1} entries in H and 2^{n-1} entries in $\overline{H}$. The matrices can be written in blocks, as follows, using proposition 13:

$$\begin{array}{c} y \\ \left(\begin{array}{c} 2^{n-1} \in H \\ \hline 2^{n-1} \in \overline{H} \end{array}\right) \end{array} \oplus \begin{array}{c} x \\ \left(\begin{array}{c} b \in H \\ \hdashline 2^{n-1} - b \in \overline{H} \\ \hline 2^{n-1} - b \in H \\ \hdashline b \in \overline{H} \end{array}\right) \end{array} = \begin{array}{c} z \\ \left(\begin{array}{c} b \in H \\ \hdashline 2^{n-1} - b \in \overline{H} \\ \hline 2^{n-1} - b \in \overline{H} \\ \hdashline b \in H \end{array}\right) \end{array}$$

where b entries from H and 2^{n-1} -b entries from $\overline{H}$ in the x matrix are added to the 2^{n-1} entries from H in the y matrix. This yields a total of 2b entries from H and 2^n - 2b entries from $\overline{H}$ in the z matrix. Since the transformation is replicative, the z column matrix has $|H| = 2^{n-1}$ entries from $\overline{H}$. Thus $2b = 2^{n-1}$ and $b = 2^{n-2} = 2^{n-1} - b$. So $|TH \cap H| = |T\overline{H} \cap \overline{H}| = 2^{n-2}$ and T is unbiased.

Q.E.D.

Proposition 14 is really included in proposition 22, but the proof of the former reveals more about the structure of the transformations and the additional property resulting from linearity.

Thus, any unbiased substitution of the n-bit binary numbers will fall into one of three categories:

1. It is replicative but not additive. Its powers are in general not replicative. It is a non-linear transformation which is also not affine.

2. It is additive with fixed point other than the identity. Its powers are also additive. This group of transformations are all affine.

3. It is additive with the identity fixed. Its powers are also additive. This group of transformations are all linear. They have the further property that they split all subgroups and not just the maximal ones.

TABLE OF CONTENTS

1. INTRODUCTION

The term Dynamic Substitution Device (DSD) was coined at Teledyne Electronics to designate a block substitution device, or S-box, in which the mappings or substitutions of clear text blocks of binary numbers to encrypted blocks of binary numbers, or vice versa, are accomplished by means of so-called *orthomorphisms.*

Any block substitution can be described in terms of look-up tables, permutations, or Boolean functions. A look-up table is simply a tabulation of corresponding clear text and cipher text pairs. A permutation is an orderly way of describing a look-up table and has the advantage of showing any cycles or subsets of numbers mapped onto themselves. Permutations also have some algebraic properties which can be considered in the design of S-boxes. Boolean functions (commonly used) are functions of n-bit binary numbers or blocks whose dependent variables are individual bits of another n-bit number. Thus, n Boolean functions are needed to describe a block substitution for n-bit numbers.

Orthomorphisms are another method of describing block substitutions, albeit a special class of them because most block substitutions are not orthomorphisms. As shown in ref. 2, there are many orthomorphisms existing. When they are described by look-up tables or Boolean functions, their special properties tend to be obscured.

Letting G_n be the group of n-bit binary numbers under coordinatewise addition modulo 2 (denoted by the symbol $\oplus$), an orthomorphism of G_n is a 1 to 1 mapping $R{:}G_n \rightarrow G_n$ such that $\{x \oplus R(x)\}_{x \epsilon G_n} = G_n$. This is equivalent to pairing each n-bit binary number with another in such a way that in the collection of 2^n pairs, each number appears exactly once in each of the two sets of numbers and in such a way that the sums modulo 2 of the pairs is again G_n. (It is not obvious that this can ever be done.) This constitutes a bijective mapping with a single fixed point since some number must be added modulo 2 to the all-zero additive identity.

2. BACKGROUND WORK WITH LINEAR AND AFFINE ORTHOMORPHISMS

Linear (automorphic) and affine orthomorphisms are described in some detail in refs. 3 and 4. A linear orthomorphism can be represented by a set of 2^n equations, for block size n, where each x_i is an n-bit binary number:

$$\begin{array}{ccccc}
\Theta & \oplus & \Theta & = & \Theta \\
x_m & \oplus & x_1 & = & x_{1-p} \\
x_1 & \oplus & x_2 & = & x_{2-p} \\
 & \cdot & & & \\
 & \cdot & & & \\
x_{k-1} & \oplus & x_k & = & x_{k-p} \\
 & \cdot & & & \\
 & \cdot & & & \\
x_{m-1} & \oplus & x_m & = & x_{m-p}
\end{array}$$

where $m = 2^n - 1$ and p is an integer determined by the generating function but independent of the initial conditions or key variable $\{x_1, x_2, \cdots, x_n\}$. This set of equations can be thought of as a set of 2^n vectors on a three-dimensional space. These vectors can be added componentwise modulo 2.

In the linear case, the set of vectors forms an additive group, but this is both a strength and a weakness. Knowing any set of n linearly independent vectors (equations) one could construct the remaining equations from the group property. As in every block substitution, the mapping can be represented by a permutation. In this case, there are three permutations depending on which pair of columns is chosen. As a convention, we typically consider the mapping to be from column 1, x_{k-1} to column 3, x_{k-p}. The compensating quality described in refs. 3 and 4 is that if column 1 is shifted with respect to column 2, column 3 is also shifted by a corresponding amount and represents a new linear orthomorphism.

The permutation representing the new orthomorphism is a power of the permutation representing the original orthomorphism. In ref. 4, it is shown that these permutations form a cyclic permutation group of order $2^n - 1$ which is transitive, *i.e.*, each number other than the fixed number is mapped on every other number by the family of orthomorphisms.

The affine orthomorphisms can be converted to or derived from linear orthomorphisms by adding vectorially to each of the 2^n original equations an equation of the form $S \oplus S = \Theta$ where S is a fixed number.

In practice, a linear DSD has been used with a continual change of shift position, or equivalently, a different member of the permutation group. Employed in this way, the S-box is nonlinear.

There is another aspect to linearity *vs.* nonlinearity, namely, the level at which it is described. Orthomorphic block substitutions have been derived by linear recursive generating functions applied to a maximal set of linearly independent n-bit numbers. The mappings of numbers from one column to another in the set of equations is also linear under the operation of modulo 2 addition bitwise; however, if the same mapping is now considered under the operation modulo m, where $m = 2^n - 1$, the mappings, in general, are nonlinear. Here, linearity is defined by the property of a mapping F, such that $F(ax + by) = aFx + bFy$ for all vectors x, y, and scalars a, b.

In ref. 1, S-boxes or mappings are defined in terms of Boolean functions, *i.e.*, a set of n functions which map an n-bit number (clear text) to one bit of an n-bit (cipher text) number. Because orthomorphisms are block substitutions, although of a special type, they can also be represented this way. In ref. 1, it is proven that a sufficient condition for the mapping or block substitution to be nonlinear at the integer level (presumably under modulo m addition) is that all n Boolean functions (at the bit level) be nonlinear. This is not claimed to be a necessary condition. In the appendix, an example is given of a linear orthomorphism (under modulo 2 addition) with linear Boolean functions which is nonlinear at the integer level modulo m.

3. NONLINEAR ORTHOMORPHISMS & DYNAMIC SUBSTITUTION DEVICES

3.1 General Considerations

In view of the comments in the last section on different definitions of nonlinearity, this section is concerned specifically with block substitutions using nonlinear orthomorphisms in the sense that they are nonlinear mappings from G_n onto G_n under the operation of bitwise addition modulo 2. The motivation to look at nonlinear orthomorphisms is twofold:

a. There are many more nonlinear than linear orthomorphisms.

b. Nonlinear orthomorphisms should be more resistant to cryptanalysis than the linear or affine versions because of the lack of group symmetries.

There is a trade-off, however, in that the nonlinear orthomorphisms do not generate an automorphism group and the corresponding dynamic substitution devices no longer can change substitutions by shifting.

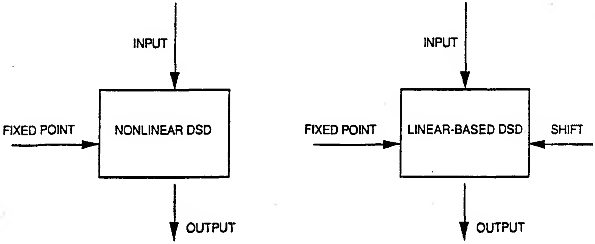


Figure 1.

The first problem is to find a way of routinely generating nonlinear orthomorphisms. In one approach, it is tempting to try a variation of the process used in ref. 4, prop. 1, relaxing the requirement that successive sets of n numbers in each column be linearly independent, while constructing a set of equations to represent an orthomorphism. Select n linearly independent numbers, and start the array of equations as follows:

$$\begin{array}{cccc} --- & \oplus & x_1 & = & --- \\ x_1 & \oplus & x_2 & = & z_2 \\ & \cdot & & & \\ & \cdot & & & \\ & \cdot & & & \\ x_{n-1} & \oplus & x_n & = & z_n \\ x_n & \oplus & \_\_\_ & = & \_\_\_ \end{array}$$

The choice of the linearly independent set $\{x_1, \cdots, x_n\}$ specifies $n - 1$ equations of the budding orthomorphism. Choosing a candidate for x_{n+1}, such that $\{x_2, \cdots, x_{n+1}\}$ and $\{z_2, \cdots, z_{n+1}\}$ are linearly independent, yields a linear or affine orthomorphism if successive steps can continue to x_m. If the restriction on linear independence is removed, the only restriction is that neither x_{n+1} nor z_{n+1} duplicate a predecessor.

The problem experienced with this procedure is that one almost always runs out of choices before reaching x_m where $m = 2^n - 1$. Without loss of generality, one can assume that the fixed point is given by the equation $\Theta \oplus \Theta = \Theta$. To avoid lengthy "cut and try" procedures, some additional guidance is needed.

Another approach is to convert a linear orthomorphism to a nonlinear one by taking the set of equations defining the linear orthomorphism and permuting the numbers in two columns in such a manner that the third column of sums is unchanged. In the next section, this approach will be explored.

3.2 Conversion of Linear Orthomorphisms to Nonlinear

Any linear orthomorphism can be expressed as an additive group of equations which can be treated as vectors and added componentwise modulo 2 to obtain another equation (vector) in the same array or group of equations representing the orthomorphic block substitution. Omitting the identity equation, $\Theta \oplus \Theta = \Theta$, the orthomorphism can be written:

$$\begin{array}{ccccc}
x_m & \oplus & x_1 & = & x_{1 \cdot p} \\
x_1 & \oplus & x_2 & = & x_{2 \cdot p} \\
 & \cdot & & & \\
 & \cdot & & & \\
 & \cdot & & & \\
x_{j-1} & \oplus & x_j & = & x_{j \cdot p} \\
 & \cdot & & & \\
 & \cdot & & & \\
 & \cdot & & & \\
x_{m-1} & \oplus & x_{n:} & = & x_{m \cdot p}
\end{array}$$

If numbers in columns 1 and 2 could be rearranged within themselves so that the sums in column 3 are preserved, a nonlinear orthomorphism would result. Since the orthomorphism is based on the relationship between three columns, it is intuitively tempting to seek some relationship among triples of rows. Because of the group structure, the vector sum of the first three equations will be another equation which already occurs at position q, where q is given by:

$$\begin{array}{ccccccc} x_1 & \oplus & x_2 & \oplus & x_3 & = & x_q \\ & & x_{2-p} & \oplus & x_3 & = & x_q \\ & & x_{3-(p+1)} & \oplus & x_3 & = & x_q \end{array}$$

This last equation is from the array representing a shift of $p + 1$ in column 1 or the $(p + 1)$st power of the basic permutation.

Thus:

$$x_{3-(p+1)} \quad \oplus \quad x_3 \quad = \quad x_{3-p_{(p+1)}}$$

and

$$q \equiv 3 - p_{(p+1)} \text{ modulo } m$$

where $p_{(p+1)}$ is the shift in column 3 corresponding to a shift of $p + 1$ in column 1. (See ref. 4.)

Without loss of generality, we take the first j rows in the linear orthomorphic array and take successive triple sums of adjacent rows. The first of these will be:

$$(x_m \oplus x_1 \oplus x_2) \quad \oplus \quad (x_1 \oplus x_2 \oplus x_3) \quad = \quad (x_{1-p} \oplus x_{2-p} \oplus x_{3-\mathrm{p}})$$

which is the row in the qth position in the array of equations and can be written:

$$x_{q-1} \quad \oplus \quad x_q \quad = \quad x_{q-p}$$

The remaining rows generated in this way are:

$$\begin{array}{ccccc} x_q & \oplus & x_{q+1} & = & x_{q+1-p} \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ x_{q+j-4} & \oplus & x_{q+j-3} & = & x_{q+j-3-p} \end{array}$$

for a total of $j - 2$ adjacent rows in the original array separated by $q - j$ rows from the set of j equations. The second group of $j - 2$ rows lend themselves to a natural transformation by cancelling the common numbers in columns 1 and 2, yielding:

$$\begin{array}{ccccc} x_m & \oplus & x_3 & = & x_{q-p} \\ x_1 & \oplus & x_4 & = & x_{q+1-p} \\ \cdot & & & & \\ \cdot & & & & \\ \cdot & & & & \\ x_{j-3} & \oplus & x_j & = & x_{q+j-3-p} \end{array}$$

In these $j - 2$ modified rows, column 1 duplicates all but x_{j-2} and x_{j-1} in column 1 of the first set and all but x_1 and x_2 from column 2. If this process is to succeed, a mechanism must be found to replace the duplicates in the first set with $j - 2$ original numbers in columns 1 and 2 of the second set. The first question, however, is whether or not this is possible.

Let S be the set of numbers in column 1 of the first j equations and T be the corresponding set of numbers in column 2. These sets must be replaced, respectively, by the sets S' and T' consisting of the original numbers in columns 1 and 2 of the second set of $j-2$ equations plus those two numbers left over in each of S and T.

$$S = \{x_m, x_1, \cdots, x_{j-1}\} \rightarrow S' = \{x_{j-2}, x_{j-1}, x_{q-1}, \cdots, x_{q+j-4}\}$$
$$T = \{x_1, x_2, \cdots, x_j\} \rightarrow T' = \{x_1, x_2, x_q, \cdots, x_{q+j-3}\}$$

The conditions on transforming from one set to another are interrelated. If $x_k \varepsilon S$ and $x_k \rightarrow x'_k \varepsilon S'$, then, $x_{k+1} \varepsilon T$ and $x_{k+1} \rightarrow x'_{k+1} \varepsilon T'$. $x_k \oplus x_{k+1} = x_{k+1-p} = x'_k \oplus '_k \; x'_{k+1}$, so that $x'_{k+1} = x_k \oplus x_{k+1} \oplus x'_k$.

Let $x_m \rightarrow x'_m \varepsilon S'$, then, from T, $x_1 \rightarrow x_m \oplus x_1 \oplus x'_m = x'_1 \varepsilon T'$. There is only one possible choice of $x'_m \varepsilon S'$ so that $x'_1 \varepsilon T'$, *i.e.*, $x'_m = x_{q-1} = x_m \oplus x_1 \oplus x_2$, in which case, $x_1 \rightarrow x'_1 = x_2$.

Similarly, at the other end, $x_{j-1} \varepsilon S$ and $x_{j-1} \rightarrow x'_{j-1} \varepsilon S'$. $x_j \varepsilon T$ and $x_j \rightarrow x'_{j-1} \oplus x_j \oplus x'_{j-1} = x'_j \varepsilon T'$. Again, there is only one possible choice for $x'_{j-1} = x_{j-2}$ so that $x'_j \varepsilon T'$, namely, $x'_j = x_{j-2} \oplus x_{j-1} \oplus x_j = x_{q+j-3} \varepsilon T'$. Thus, the first and last modified rows of the first set of j equations are uniquely specified:

$$x_m \oplus x_1 = x_{1-p} \text{ becomes } x_{q-1} \oplus x_2 = x_{1-p}$$

and:

$$x_{j-1} \oplus x_j = x_{j-p} \text{ becomes } x_{j-2} \oplus x_{q+j-3} = x_{j-p}.$$

Next, consider the second original equation:

$$x_1 \oplus x_2 = x_{2-p}$$

$x_1 \rightarrow x'_1 \varepsilon S'$. $x'_1 \neq x_{q-1}$ or x_{j-2} since these have been used. $x_2 \rightarrow x'_2 \varepsilon T'$ where $x'_2 = x_1 \oplus x_2 \oplus x'_1$

If $x'_1 = x_q, x_{q+1}, \cdots, x_{q+j-4}$, then, $x'_2 \notin T'$. So, the only possibility is $x'_1 = x_{j-1} \varepsilon S'$. In that case, $x'_2 = x_1 \oplus x_2 \oplus x_{j-1}$. $j \geq 3$:

$$\begin{array}{llll} \text{if } j = 3 & x'_1 = x_2 & \text{and } \; x'_2 = x_1 & x_2 \oplus x_1 = x_{2-p}\,. \\ \text{if } j = 4 & x'_1 = x_3 & \text{and } \; x'_2 = x_q & x_3 \oplus x_q = x_{2-p}\,. \\ \text{if } j > 4 & x'_2 \notin T'\,. & & \end{array}$$

If $j = 4$, there is one more equation of the first set of four to be modified: the third. Because $j = 4$, $S' = \{x_2, x_3, x_{q-1}, x_q\}$ and $T' = \{x_1, x_2, x_q, x_{q+1}\}$. The only unused pair is $x_q \varepsilon S'$ and $x_1 \varepsilon T'$. Fortunately, $x_q \oplus x_1 = x_{3-p}$, and the transformation is complete.

3.3 Nonlinearization Summary

Nonlinearization by taking sums of consecutive triples of rows works, if and only if, a set of three or four consecutive rows is used. If three consecutive rows are selected, the result is:

	Original	Modified
1.	$x_m \oplus x_1 = x_{1\text{-}p}$	$x_{q\text{-}1} \oplus x_2 = x_{1\text{-}p}$
2.	$x_1 \oplus x_2 = x_{2\text{-}p}$	$x_2 \oplus x_1 = x_{2\text{-}p}$
3.	$x_2 \oplus x_3 = x_{3\text{-}p}$	$x_1 \oplus x_q = x_{3\text{-}p}$
q	$x_{q\text{-}1} \oplus x_q = x_{q\text{-}p}$	$x_m \oplus x_3 = x_{q\text{-}p}$

The modification can be obtained by adding vectorially to each of the four rows:

$$(x_1 \oplus x_2) \oplus (x_1 \oplus x_2) = \Theta$$

If four consecutive rows are selected, the result is:

	Original	Modified
1.	$x_m \oplus x_1 = x_{1\text{-}p}$	$x_{q\text{-}1} \oplus x_2 = x_{1\text{-}p}$
2.	$x_1 \oplus x_2 = x_{2\text{-}p}$	$x_3 \oplus x_q = x_{2\text{-}p}$
3.	$x_2 \oplus x_3 = x_{3\text{-}p}$	$x_q \oplus x_1 = x_{3\text{-}p}$
4.	$x_3 \oplus x_4 = x_{4\text{-}p}$	$x_2 \oplus x_{q+1} = x_{4\text{-}p}$
q	$x_{q\text{-}1} \oplus x_q = x_{q\text{-}p}$	$x_m \oplus x_3 = x_{q\text{-}p}$
$q+1$	$x_q \oplus x_{q+1} = x_{q+1\text{-}p}$	$x_1 \oplus x_4 = x_{q+1\text{-}p}$

The modification can be obtained by adding vectorially:

to rows 1 and q	$(x_1 \oplus x_2) \oplus (x_1 \oplus x_2) = \Theta$
to rows 2 and 3	$(x_1 \oplus x_3) \oplus (x_1 \oplus x_3) = \Theta$
to rows 4 and $q+1$	$(x_2 \oplus x_3) \oplus (x_2 \oplus x_3) = \Theta$

3.4 Application of Nonlinearization Techniques

It is often tempting to try to speak of levels of nonlinearity as if such levels were measurable quantities. Somehow, piecewise linear functions seem less nonlinear than transcendental functions. In cryptography, one refers to the *Hamming distance* of nonlinear Boolean functions from linear ones. While linearity is well defined, and nonlinear functions are simply everything left over when the linear, and perhaps affine functions are deleted, in what follows we will succumb to the tendency to ascribe measurability to levels of nonlinearity.

In the last section, two means were described to nonlinearize four or six rows at a time of an arbitrary linear orthomorphism. Because those methods could be applied, starting with any row in the orthomorphic array, each could yield $2^n - 1$ different, nonlinear orthomorphisms from one linear orthomorphism; however, they would not "be very nonlinear" in the sense that for large n, a big chunk of the original array of equations would be left intact. So it is natural to use one or both of these processes repeatedly. The limitation, short of running out of rows, is having chunks of rows overlap. The fraction of original rows, modified, might be considered a measure of nonlinearity, so the next question is how to select efficiently clumps of non-overlapping rows.

3.5. Selection of Rows or Equations for Nonlinearization

There are many ways to choose sets of triples or quadruples of consecutive equations for this nonlinearization process. The only essential criterion is that no rows derived as triple sums from some triple or quadruple sets overlap with another triple or quadruple set. In addition to this, it seems obvious that as many rows as possible from the linear orthomorphism be modified and with a reasonably uniform distribution so as to avoid any piecewise linear portions in the new orthomorphism. First, consider the case in which $m = 2^n - 1$ is a composite number. Let d be a factor of m. From the linear orthomorphisms, select rows number 1, 2, 3 and 4 along with the corresponding sum rows q and $q + 1$. We can then select successive sextuples of rows to be modified as in the preceding section. The row numbers of these successive sextuples will be:

$$\begin{array}{cccccc}
1 & 2 & 3 & 4 & q & q+1 \\
1+d & 2+d & 3+d & 4+d & q+d & q+1+d \\
 & \cdot & & & \cdot & \\
 & \cdot & & & \cdot & \\
 & \cdot & & & \cdot & \\
1+ad & 2+ad & 3+ad & 4+ad & \mathrm{q}+ad & q+1+ad
\end{array}$$

where a takes on successive values 0, 1, 2, ···. This process can terminate in two ways. First of all, one of the equations in a quadruple set may be duplicated in another set, *i.e.*, overlapping quadruplets and the same may happen between the sets of equations which are the triple sums. This will occur when for some integer a:

$$1 + ad, 2 + ad, 3 + ad \text{ or } 4 + ad \equiv 1, 2, 3 \text{ or } 4 \text{ modulo } m$$

$$q + ad \text{ or } q + 1 + ad \equiv q \text{ or } q + 1 \text{ modulo } m.$$

This is equivalent to:

$$ad \equiv 0, \pm 1, \pm 2 \text{ or } \pm 3 \text{ modulo } m.$$

The minimum separation between successive initial rows, *i.e.*, 1 and $1 + d$, must be at at least 4 for quadruples. Since $m = 2^n - 1$ is odd, and d divides m, d is also odd and, thus, $d \geq 5$. this also means that $ad \not\equiv \pm1, \pm2, \pm3$ modulo m.

The second means of potential overlap is when:

$$1, 2, 3, \text{ or } 4 \equiv q + ad \text{ or } q + 1 + ad \text{ modulo } m$$

$$1 + ad, 2 + ad, 3 + ad, \text{ or } 4 + ad \equiv q \text{ or } q + 1 \text{ modulo } m.$$

This is equivalent to:

$$ad \equiv \pm q, \pm(q - 1), \pm(q - 2), \pm(q - 3), \text{ or } \pm(q - 4) \text{ modulo } m.$$

To avoid this, d must be relatively prime to $q, \cdots, q - 4$; however, one of these numbers must be a multiple of 5 so that we require $d \geq 7$. If triples rather than quadruples of rows are selected, then the conditions $ad \not\equiv \pm q$ or $\pm(q - 4)$ are eliminated. In that case, $d \geq 5$ satisfies the conditions above. (*Note:* All integers here are positive, so the condition $ad \not\equiv -q$ modulo m, *etc*, means $ad \not\equiv bm - q$. Since d divides m, if it is relatively prime to q, it must also be relatively prime to $bm - q$ for any integer b.)

3.5 Nonlinearization Techniques Summary

If $m = 2^n - 1$ is not a prime, seek a factor d of m, relatively prime to q, $q - 1, \cdots, q - 4$ to space successive quadruples of equations (relatively prime to $q - 1$, $q - 2$, $q - 3$ to space successive triples of equations) for the nonlinearization process.

When such a number d can be found, it leads a uniformly spaced, dense set of equations in the linear array for nonlinearization. Some examples are given in the appendix.

3.6 Other Selections of Rows

When the above method does not appear to be rewarding, another obvious possibility is to take successive quadruples of rows, defined by a triple and its sum, in which the first row of a subsequent quadruple follows immediately after the last row (triple sum) of the previous quadruple. For example:

1	2	3	q
$q + 1$	$q + 2$	$q + 3$	$2q$
	.		.
	.		.
	.		.
$bq + 1$	$bq + 2$	$bq + 3$	$(b + 1)q$

This process terminates when any one of three conditions occurs:

1. $bq \equiv 0, \pm 1, \pm 2$ modulo m
2. $(b-1)q \equiv -1, -2, -3$ modulo m
3. $(b+1) \equiv 1, 2, 3$ modulo m

If q is a prime, or at least not a divisor of $m \pm 1$, $m \pm 2$, or $m \pm 3$, this process can yield a large number of nonoverlapping quadruples for nonlinearization. This process can also be done with sextuples of rows, but the conditions for termination are more complex.

In general, these two methods, under proper conditions, can select nonoverlapping sets of equations from the linear orthomorphism for nonlinearization. Only a relatively small number of equations, evenly distributed, are left unmodified.

There are many other ways to go about this process, *e.g.*, fitting sets of quadruples and sextuples of rows together without overlapping. This becomes a topological problem. The important point is that it is possible in this way to generate nonlinear orthomorphisms which are not piecewise linear.

4. MEASURES OF CRYPTOGRAPHIC BIAS

The author is unaware of any generally accepted measurable standard for freedom from bias in block substitutions of binary numbers. Two widely used criteria are *avalanching* (good or strict) and *bit independence*. In what follows, two other possible criteria are suggested: (1) balance, which is related to properties of the n-bit binary numbers as an algebraic group, and (2) transitivity, which is a property of certain permutation groups that in not commonly applied to block substitutions. Of the four, it appears to be possible to meet, at most, three criteria simultaneously.

4.1 Avalanching and Bit Independence

The strict avalanche criterion (SAC) is defined as the property that changing the ith bit in the input block causes a change in the jth bit in the output block one half of the time for all $1 \leq i, j \leq n$, where n is the block size. The bit independence criterion (BIC) is defined as the property that changing the ith bit in the input block of the averages changes half the bits in the output block. (See ref. 1.) These properties can also be described in terms of the algebraic properties of the binary numbers of arbitrary block size.

G_n is the additive group of all n-bit binary numbers, where the symbol $\ominus$ means modulo 2 addition bitwise.

There are n maximal subgroups H_i which may be termed "basic" which are defined by the property that each block in H_i has 0 in the ith position. Numbering from left to right, H_n consists of all even numbers, and H_1, consists of all lower half numbers. In general:

$$H_i = \{x \,\varepsilon\, G_n \mid x = \underset{1}{-} - \cdots - \underset{i}{0} - - \cdots \underset{n}{-}\}$$

Clearly, H_i is characterized by the property that:

$$x_i H_i = 0 \text{ where } x_i = \underset{1}{0} \cdots 0 \underset{i}{1} 0 \cdots \underset{n}{0}$$

$$|H_i| = \frac{1}{2}|G_n| = 2^{n-1} .$$

There are a total of $2^n - 1$ maximal subgroups (MSG) which can be generated from these basic MSG's by the operation:

$$H_k = H_i + H_j \equiv (H_i \cap H_j) \cup (\bar{H}_i \cap \bar{H}_j)$$

This is somewhat analogous to the Boolean rings described in ref. 2, section 9.4; however, the multiplicative semi-groups property defined by the intersection of sets does not apply here because the intersections of MSG's are not MSG's. The addition operation is actually the complement of that defined above, which would, in effect, define the sum of two MSG's to be the complement of another.

Changing the ith bit in some number x to obtain a new number y is equivalent to choosing y such $x \oplus y = 0\ 0 \cdots 0\ \underset{i}{1}\ 0 \cdots 0$. For a given i there are 2^{n-1} such pairs. SAC is equivalent to requiring that for $1 \leq i, j \leq n$ $Tx \oplus Ty$ has 0 as the jth bit one half of the time (no change) and 1 as the jth n-bit one half of the time (change). More formally, define:

$$P_i = \{Tx \oplus Ty \mid x \oplus y = 0 \cdots 0\ 1\ 0 \cdots 0\} \text{ where 1 is at the } i\text{th bit position.}$$

Note: This implies that $x \,\varepsilon\, H_i$ and $y \,\varepsilon\, \bar{H}_i$ or vice versa. $|P_i| \leq 2^{n-1}$ since there are 2^{n-1} such pairs but the sums may not be distinct. Thus, SAC is equivalent to:

$$|P_i \cap H_j| = |P_i \cap \bar{H}_j| = \frac{1}{2}|P_i| \text{ for } 1 \leq i, j \leq n \quad n^2 \text{ conditions.}$$

P_i consists of 2^{n-1} n-bit numbers which may range from all the same to all different. Each bit results from adding a (unordered) pair (0,1), (0,0), (1,1). If the number of such pairs is, respectively, a, b, c, then $a + b + c = 2^{n-1}$, the number of pairs defining P_i. The number of zeroes in the pairs is $a + 2b = 2^{n-1}$ and the number of one's is $a + 2c = 2^{n-1}$; therefore, a is even and the number of 1's and 0's appearing in each bit position of the 2^{n-1} numbers in P_i is even. For example, if $n = 4$, P_i will contain eight 4-bit numbers, perhaps not distinct but each bit position will contain 0, 2, 4, 6 or 8 one's. If they are not evenly divided (avalanching), the closest approximation will be 2:6.

Considering P_i a set of 2^{n-1} numbers, including multiple occurrences, a measure of approximate avalanching is $\frac{|P_i \cap H_j|}{2^{n-1}}$ averaged over all n^2 i, j pairs.

The bit independence criterion is related to avalanching. Consider the set P_i including the multiplicity of each member. To meet BIC, the number of zeroes and ones should be approximately equal in each block. For n odd, of course, this can never occur in an individual block. There are 2^{n-1} n-bit numbers (including duplicates) in P_i with a total of $n2^{n-1}$ binary bits. Let NP_i be the number of zeroes in the set P_i, including duplicates or multiple appearances. $\frac{NP_i}{n2^{n-1}}$ is a measure of BIC for a given bit position. Overall, BIC specifies that:

$$\frac{\sum\limits_{i=1}^{n} NP_i}{n^2 2^{n-1}} = \frac{1}{2}$$

For a given P_i, let the numbers be enumerated as follows, where Z_{ij} is the jth position of the ith number and $l = 2^{n-1}$.

$$\begin{array}{cccc}
Z_{11} & Z_{12} & \cdots\cdots & Z_{1n} \\
 & \cdot & & \\
 & \cdot & & \\
 & \cdot & & \\
Z_{i1} & Z_{i2} & \cdots\cdots & Z_{im} \\
 & \cdot & & \\
 & \cdot & & \\
 & \cdot & & \\
Z_{l1} & Z_{l2} & \cdots\cdots & Z_{ln}
\end{array}$$

Then, avalanching specifies that the zeroes and ones in each column will be approximately evenly divided, and BIC specifies the equivalent condition on the rows.

Both SAC and BIC are usually referred to as bit level properties because they refer to relations between individual bits in the same and different n-bit numbers for a given S-box or mapping. Thus, if applied to orthomorphic mappings, these properties depend on the key variable or the particular set of n linearly independent numbers acted on by the generating functions.

4-2. Other Criteria for Lack of Bias

The previous section briefly described the maximal subgroups (MSG's) of the group G_n of n-bit binary numbers where the group operation ($\oplus$) is bitwise addition modulo 2. There are $m = 2^n - 1$ such MSG's $H_1, \cdots, H_m$ which themselves form a group under the operation of Boolean addition:

$$H_k = H_i + H_j \equiv (H_i \cap H_j) \cup (\bar{H}_i \cap \bar{H}_j)$$

The n MSG's which are characterized by all having zero in a given bit position, *e.g.*, the even numbers, are easy to visualize conceptually. They are implicitly used in the definition of avalanching; however, the other $m - n$ MSG's are of equal significance algebraically, and any n linearly independent MSG's can be used to generate the entire set. In ref. 3, the following was introduced:

Definition: A bijective mapping on G_n is said to be balanced if it maps each maximal subgroup, *i.e.*, each subgroup of order $2^n - 1$, half into itself and half into its complement.

If T is the bijective mapping, this means $|TH_i \cap H_i| = |TH_i \cap \bar{H}_i| = 2^{n-2}$ $1 \le i \le 2^n - 1$. It was also shown in ref. 3 that T is an orthomorphism if and only if it is balanced. Thus, balance is a unique and universal property of orthomorphic block substitutions at the integer level, independent of linearity or key variable. A practical consequence is that in design of an orthomorphic S-box, no special effort need be made to incorporate or verify this property.

Transitivity is not an obvious quality to attribute to S-boxes since it is a property of certain permutation groups. If we are considering permutations on $m = 2^n - 1$ numbers (or letters) of which block substitutions are an example, then the definition from ref. 6, pg. 82, is pertinent:

Definition: A group of permutations is said to be transitive if it contains at least one permutation which transforms any one of the m letters (numbers) into any other letter (number).

However, by prop. 6 of ref. 3, linear orthomorphisms form a transitive group on the $2^n - 1$ nonfixed numbers. In fact, this is also true for affine orthomorphisms. This property, which implies a thorough mixing at the integer level is, of course, a property not of a single block substitution but of a group of them.

The collection of all possible block substitutions or bijective mappings on G_n, viewed as permutations, constitute a transitive group, namely P_m, the symmetric group on m elements and of order $m!$. Other transitive subgroups can be generated by taking all powers of a maximal permutation, that is, one with no proper subcycles.

In the case of linear or automorphic orthomorphisms, this is particularly easy because of the property of being able to shift columns in the array of equations defining the automorphisms. Maximal nonlinear orthomorphisms could be used to generate a transitive subgroup but, in general, the other members would not be orthomorphisms.

4-3. Applications to Orthomorphisms

If T is a linear (automorphic) or affine orthomorphism, it is clear that strict avalanching is not possible. In this case, $Tx \oplus Ty = T(x \oplus y) + S$ where S is a fixed number ($S = 00 \cdots 0$ if T is linear). Thus, P_i consists of a single number and $|P_i \cap H_j| = 0$ or 1 and $|P_i \cap \bar{H}_j| = 1$ or 0. However, the n numbers P_i may have an equal distribution of zeros and ones so that bit independence is possible.

On the other hand, the nonlinear orthomorphisms constructed from linear ones have shown good avalanching and BIC. An example is given in the appendix of a nonlinear orthomorphism with strict avalanching and a BIC ratio of exactly $\frac{1}{2}$.

All orthomorphisms and only orthomorphisms have balance as defined in ref. 3. Linear and affine orthomorphisms are represented by transitive groups of permutations. Thus, over the family of block substitutions, each clear text block is mapped to each possible cipher text block except for the fixed point which may be varied. This can be summarized as follows:

a. SAC and BIC: Bit level properties depending on the key variable

b. Balance: A universal integer level property of all orthomorphisms

c. Transitivity: A universal integer level property for groups of linear or affine orthomorphisms and also:

	Orthomorphisms		Other
Property	Nonlinear	Linear/Affine	Substitutions
Avalanching	Yes	No	Yes
Bit independence	Yes	Yes	Yes
Balance	Yes	Yes	No
Transitivity	No	Yes	Possibly[*]

[*] This is a group property and depends on selecting a block substitution which is maximal, i.e., no cycles, and then taking all powers of it.

6. REFERENCES

1. "The Structured Design of Cryptographically Good S-Boxes," Adams and Tavares, J, Cryptology (1990) 3:27-41

2. "Fundamentals of Mathematics," Vol 1, Behnke, Bachman, *et. al.*, MIT Press 1986.

6. "Introduction to the Theory of Finite Groups," W. Ledermann, Oliver and Boyd, 1964.

7. "Shift Register Sequences," S.W. Golomb, Aegean Park Press, 1982.

APPENDIX

A. LINEAR ORTHOMORPHISMS VIEWED FROM VARIOUS PERSPECTIVES

As pointed out in section 2, a linear orthomorphism can be viewed and represented in several ways. For example, consider $n = 4$, with generating function $x_k = x_{k-4} \oplus x_{k-3}$ for which $p = 12$, and with base set or linearly independent set:

$$x_1 = 0001, x_2 = 0010, x_3 = 0100, x_4 = 1000.$$

This defines the table:

Column 1		Column 2		Column 3
0000	⊕	0000	=	0000
1001	⊕	0001	=	1000
0001	⊕	0010	=	0011
0010	⊕	0100	=	0110
0100	⊕	1000	=	1100
1000	⊕	0011	=	1011
0011	⊕	0110	=	0101
0110	⊕	1100	=	1010
1100	⊕	1011	=	0111
1011	⊕	0101	=	1110
0101	⊕	1010	=	1111
1010	⊕	0111	=	1101
0111	⊕	1110	=	1001
1110	⊕	1111	=	0001
1111	⊕	1101	=	0010
1101	⊕	1001	=	0100

It is easily verified that the mapping from column 1 to column 3 (or any other pair of columns) is linear under addition modulo 2. That same mapping could be represented by a table in which y (column 3) is a function $f(x)$ of x (column 1). In decimal notation this becomes:

x		y
0	→	0
1		3
2		6
3		5
4		12
5		15
6		10
7		9
8		11
9		8
10		13
11		14
12		7
13		4
14		1
15		2

It is easy to see that this is nonlinear under addition modulo 16 or by representing it in graphical form. The mapping can also be represented as a permutation:

$$(0)(1, 3, 5, 15, 2, 6, 10, 13, 4, 12, 7, 9, 8, 11, 14)$$

It can also be written in terms of Boolean functions by rewriting column 1 in the natural order of the numbers and correspondingly rearranging column 3. The Boolean functions describe bits in column 3 as functions of blocks in column 1.

Column 1			
b_1	b_2	b_3	b_4
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Column 3			
f_1	f_2	f_3	f_4
0	0	0	0
0	0	1	1
0	1	1	0
0	1	0	1
1	1	0	0
1	1	1	1
1	0	1	0
1	0	0	1
1	0	1	1
1	0	0	0
1	1	0	1
1	1	1	0
0	1	1	1
0	1	0	0
0	0	0	1
0	0	1	0

From this, the Boolean functions can be written:

$$f_1 = b_1 \oplus b_2$$
$$f_2 = b_2 \oplus b_3$$
$$f_3 = b_1 \oplus b_3 \oplus b_4$$
$$f_4 = b_1 \oplus b_4$$

The generating function $x_k = x_{k\text{-}4} \oplus x_{k\text{-}3}$ operates universally on any set of 4 linearly independent 4-bit numbers; however, the Boolean functions depend on the specific mapping. For example, consider the same generating function but applied to the base:

$$x_1 = 0011, x_2 = 0110, x_3 = 1100, x_4 = 1000$$

This defines a new table:

Column 1		Column 2		Column 3
0000	⊕	0000	=	0000
1011	⊕	0011	=	1000
0011	⊕	0110	=	0101
0110	⊕	1100	=	1010
1100	⊕	1000	=	0100
1000	⊕	0101	=	1101
0101	⊕	1010	=	1111
1010	⊕	0100	=	1110
0100	⊕	1101	=	1001
1101	⊕	1111	=	0010
1111	⊕	1110	=	0001
1110	⊕	1001	=	0111
1001	⊕	0010	=	1011
0010	⊕	0001	=	0011
0001	⊕	0111	=	0110
0111	⊕	1011	=	1100

As in the first example, to obtain the Boolean functions, it is convenient to rearrange columns 1 and 3:

Column 1				Column 3			
b_1	b_2	b_3	b_4	f_1	f_2	f_3	f_4
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	1
0	1	0	0	1	0	0	1
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	1	0	1	1
1	0	1	0	1	1	1	0
1	0	1	1	1	0	0	0
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	0
1	1	1	0	0	1	1	1
1	1	1	1	0	0	0	1

From this, the Boolean functions can be written:

$$f_1 = b_1 \oplus b_2$$
$$f_2 = b_1 \oplus b_4$$
$$f_3 = b_3 \oplus b_4$$
$$f_4 = b_1 \oplus b_2 \oplus b_3$$

Except for f_1, these differ from the previous example.

B. SELECTION OF EQUATIONS OR ROWS IN AN AUTOMORPHIC (LINEAR) ORTHOMORPHISM FOR NONLINEARIZATION

Section 3.5 suggested various ways to choose nonoverlapping quadruples or sextuples of equations for nonlinearization. The idea is to nonlinearize a large fraction of the $m = 2^n - 1$ equations in the array which defines the orthomorphic mapping. Since $2^n - 1$ is odd, it will never be possible by the methods of section 3.2 to modify all the equations because the process is applied to sets with four or six members. There is no optimal process for all block sizes or even for different generating functions with the same block size. Following are some examples for block sizes $n = 4, 5, 6, 7$, and 8.

For $n = 4$, there is just one pair of conjugate generating functions with $p = 4$, $q = 8$ and $p = 12$, $q = 11$, respectively. The corresponding maximal density of rows is given by:

1	2	3	4	8	9
5	6	7		12	

and

1	2	3	4	11	12
5	6	7		15	

for a modification of 10 of the 15 rows.

For $n = 5$, the optimum again is a hybrid mix. For example, for $x_k = x_{k-5} \oplus x_{k-2}$, $p = 18$, $q = 23$, and $m = 31$. Twenty-four rows out of 31 can be modified, selecting:

1	2	3	4	23	24
7	8	9	10	29	30
14	15	16		5	
20	21	22		11	
25	26	27		6	

For $p = 19$, $q = 11$ corresponding to $x_k = x_{k-5} \oplus x_{k-4} \oplus x_{k-2} \oplus x_{k-1}$ one can do a little better with 28 rows out of 31 for modification by selecting:

1	2	3	4	11	12
7	8	9	10	17	18
13	14	15	16	23	24
19	20	21	22	29	30
26	27	28		5	

For $n = 6$ and $x_k = x_{k-6} \oplus x_{k-1}$, the shift $p = 6$ and $q = 40$. Since $m = 63$ of which 7 is a factor, $d = 7$ is a candidate for a spacing as described in section 3.5. Also, for $q = 40$, 7 is not a divisor of $q, q - 1, \cdots, q - 4$. This yields 54 out of 63 rows for modification as follows:

1	2	3	4	40	41
		⋮			
$1 + a7$	$2 + a7$	$3 + a7$	$4 + a7$	$40 + a7$	$41 + a7$
		⋮			
57	58	59	60	33	34

Once again, the selection of sets of equations or rows for modification depends on the generating function. This particular selection will not work for most other values of q corresponding to $n = 6$.

For $n = 7$, $m = 127$ is a Mersenne prime, so the previous shortcut is not available; however, for certain generating functions, one can do quite well. Consider $n = 7, x_k = x_{k-7} \oplus x_{k-5} \oplus x_{k-4} \oplus x_{k-3} \oplus x_{k-2} \oplus x_{k-1}$ for which $p = 19$ and $q = 9$. Take rows as follows:

1	2	3		9	

then,

5	6	7	8	13	14
10	11	12		18	
15	16	17		23	

Following the same pattern of a sextuple followed by two quadruples, through:

103	104	105	106	111	112
108	109	110		116	
113	114	115		121	

and then completing with rows:

118	119	120		126
123	124	125		4

124 out of 127 equations can thus be modified, omitting only 117, 122, and 127. Other variations on this will work just as well omitting only three equations.

Also for $n = 7$, $x_k = x_{k-7} \oplus x_{k-6} \oplus x_{k-4} \oplus x_{k-2}$, $p = 107$ and $q = 10$, an analogous system works taking two sextuples and one quadruple followed by adjustments among the last rows. Again, 124 out of 127 equations can be selected for modification; however, this density or efficiency does not appear to hold for all generating functions.

For $n = 8$, consider the generating function $x_k = x_{k-8} \oplus x_{k-6} \oplus x_{k-5} \oplus x_{k-3}$ for which $p = 16$, $q = 54$, and $m = 2^n - 1 = 255$. We will select quadruples of rows consisting of successive triples, starting with the first row other than the identity, plus the row which is the triple sum of the first three. Referring to section 3.5, choose the row spacing $d = 5$. 5 is a factor of m and relatively prime to $q - 1 = 53$, $q - 2 = 52$, and $q - 3 = 51$. This will lead to selection of the following rows:

$$\begin{array}{ccccc}
1 & 2 & 3 & & 54 \\
6 & 7 & 8 & & 59 \\
 & & \vdots & \vdots & \\
1 + 5a & 2 + 5a & 3 + 5a & & 54 + 5a \\
 & & \vdots & \vdots & \\
251 & 252 & 253 & & 49
\end{array}$$

where $0 \leq a \leq 50$. This selects 51 X 4 = 204 rows or equations for nonlinearization from the total of 255 in the orthomorphic array. Left out are row 5 and all multiples of 5.

This spacing, however, will not work efficiently if we choose sextuples of rows because $q - 4 = 50$, a multiple of 5. In fact, when $a = 10$, $4 + 10a = 54$, so that the process will terminate with just 6 X 10 = 60 rows. However, there are several ways in which the remaining 195 rows can be divided into quadruples or sextuples, more or less piecemeal.

The spacing $d = 17$ is also a divisor of $m = 255$, but it is also a divisor of $q - 3 = 51$ and, hence, is not an efficient spacing.

Again, for $n = 8$ but with the generating function $x_k = x_{k-8} \oplus x_{k-4} \oplus x_{k-3} \oplus x_{k-2}$ for which $p = 25$, $q = 60$, and $m = 255$, $d = 17$ is a factor of m but now it is not a divisor of $q = 60$, $q - 1 = 59$, ····, $q - 4 = 56$. Sextuples of rows can be selected as follows:

1	2	3	4	60	61
18	19	20	21	77	78
	⋮		⋮		
1 + 17*a*	2 + 17*a*	3 + 17*a*	4 + 17*a*	60 + 17*a*	61 + 17*a*

This process terminates when $a = 14$, for a total of 15 X 6 = 90 rows. For $a = 15$, $1 + 17a = 256 \equiv 1$ modulo m, and, so, the process repeats; however, one can start again as follows:

5	6	7	8	64	65
22	23	24	25	81	82
	⋮		⋮		
5 + 17*a*	6 + 17*a*	8 + 17*a*	9 + 17*a*	64 + 17*a*	65 + 17*a*

This process again terminates when $a = 14$ for 15 X 6 = 90 additional rows. Again, at $a = 15$ the process repeats. This gives a total of 180 rows. None of the remaining 75 rows contain triples with corresponding sums.

However, as in $p = 16$, $q = 54$, quadruples can be selected with a spacing of $d = 5$ again because 5 is relatively prime to $q - 1 = 59$, $q - 2 = 58$, and $q - 3 = 57$.

This is easy to apply for suitable combinations of m and q, that is, for cooperative generating functions; however, it is not obvious how many equations representing the linear orthomorphism must be modified to get a "good" nonlinear mapping.

C. THE DEPENDENCE OF AVALANCHING AND BIT INDEPENDENCE ON SYSTEM PARAMETERS

In section 4-1 it was pointed out that avalanching and bit independence depend on the actual numbers in the block substitution and not just the algebraic structure. Ti illustrate this, we take an $n = 4$ bit family of block substitutions defined by a linear orthomorphism with generating function $x_k = x_{k-4} \oplus x_{k-3}$ for which the shift $p = 12$ and $q = 11$.

As the first example, we generate a nonlinear orthomorphism using rows:

1	2	3	11
5	6	7	15

and with base or linearly independent set $x_1 = 0001, x_2 = 0010, x_3 = 0100, x_4 = 1000$. This is a nonlinearized version of the first linear orthomorphic mapping in section A of this appendix. As modified, it becomes:

0000	⊕	0000	=	0000
1010	⊕	0010	=	1000
0010	⊕	0001	=	0011
0001	⊕	0111	=	0110
0111	⊕	1110	=	1001
1110	⊕	1111	=	0001
1111	⊕	1101	=	0010
1101	⊕	0110	=	1011
0110	⊕	0011	=	0101
0011	⊕	1001	=	1010
1001	⊕	0100	=	1101
0100	⊕	1000	=	1100
1000	⊕	1100	=	0100
1100	⊕	1011	=	0111
1011	⊕	0101	=	1110
0101	⊕	1010	=	1111

To measure avalanching and bit independence as described in section 4-1, we first enumerate the set $P_1 = \{Tx \oplus Ty \mid x \oplus y = 1000\}$, that is, pairs x, y which differ only in the first bit position. For $n = 4$, the sixteen 4-bit number pairs form eight pairs, not necessarily distinct. So, including multiplicity of appearance, $|P_1| = |P_2| = |P_3| = |P_4| = 8$. Also recall that H_i is the maximal subgroup of n-bit numbers with 0 in the ith bit position. Here, for $n = 4$, H_4 is the subgroup of even numbers, and H_1 is the subgroup of lower half numbers, *etc*. For P_1, the tabulation is on the following worksheet. Each $|P_1 \cap H_j| = 4$ which means that 4 out of 8, or one half of the bits in each member of P_1 is 0 in the jth bit position, *i.e.*, in the columnar tabulation, each column is comprised of half 0's and half 1's. The number of 0's in each member of P_1 is tabulated, *i.e.*, horizontal tabulation. The total number of 0's, $NP_1 = 16$ out of 32 bits, so that half are 0's.

When the same tabulation is made for $P_1, P_3, P_4,$ it turns out that $|P_1 \cap H_j| = 4$ for all 16 pairs, $NP_1 = 16$ for each p_1 so that the avalanching is strict and

$$\frac{\sum NP_i}{n^2 2^{n-1}} = \frac{64}{128} = \frac{1}{2}$$

WORKSHEET
COMPUTATION OF AVALANCHING AND BIT INDEPENDENCE

	Clear Bits					Cipher Bits				$P_1 = \{Tx \oplus Ty\}$				
	1	2	3	4		1	2	3	4	1	2	3	4	No. of 0 Bits
x	0	0	0	0	Tx	0	0	0	0	0	1	0	0	3
y	1	0	0	0	Ty	0	1	0	0					
	0	0	0	1		0	1	1	0	1	0	1	1	1
	1	0	0	1		1	1	0	1					
	0	0	1	0		0	0	1	1	1	0	1	1	1
	1	0	1	0		1	0	0	0					
	0	0	1	1		1	0	1	0	0	1	0	0	3
	1	0	1	1		1	1	1	0					
	0	1	0	0		1	1	0	0	1	0	1	1	1
	1	1	0	0		0	1	1	1					
	0	1	0	1		1	1	1	1	0	1	0	0	3
	1	1	0	1		1	0	1	1					
	0	1	1	0		0	1	0	1	0	1	0	0	3
	1	1	1	0		0	0	0	1					
	0	1	1	1		1	0	0	1	1	0	1	1	1
	1	1	1	1		0	0	1	0					

$|P_1 \cap H_1| = 4$

$|P_1 \cap H_2| = 4$

$|P_1 \cap H_3| = 4$

$|P_1 \cap H_4| = 4$

$NP_1 = 16$

For comparison, we can now use the second example in section A, nonlinearized with the same set of equations, and differing only in that a new base set is used:

$$x_1 = 0011, x_2 = 0110, x_3 = 1100, x_4 = 1000$$

This second nonlinear orthomorphism can be written as follows:

0000	⊕	0000	=	0000
1110	⊕	0110	=	1000
0110	⊕	0011	=	0101
0011	⊕	1001	=	1010
1001	⊕	0010	=	1011
0010	⊕	0001	=	0011
0001	⊕	0111	=	0110
0111	⊕	1010	=	1101
1010	⊕	0101	=	1111
0101	⊕	1011	=	1110
1011	⊕	1100	=	0111
1100	⊕	1000	=	0100
1000	⊕	0100	=	1100
0100	⊕	1101	=	1001
1101	⊕	1111	=	0010
1111	⊕	1110	=	0001

Making the same computation as on the sample worksheet, the results are as follows:

$$|P_1 \cap H_1| = |P_1 \cap H_2| = 0, \; |P_1 \cap H_3| = 8$$

and $|P_i \cap H_j| = 4$ for the remaining 13. $NP_1 = 12$, and $NP_2 = NP_3 = NP_4 = 16$. Thus:

$$\frac{\sum NP_i}{128} = \frac{60}{128} = 0.47$$

The avalanching is no longer strict but since almost half the time there is a bit change, it might be called "good".

For the third example, we again return to the same linear orthomorphism with $p = 12$ and $q = 11$, but this time nonlinearize it with a partially different collection of rows, that is, a sextuple and a quadruple:

1	2	3	4	11	12
13	14	15		8	

However, the same base or linearly independent set is used as in the first example, that is, $x_1 = 0001, x_2 = 0010, x_3 = 0100, x_4 = 1000$. The third nonlinear orthomorphism can be written as follows:

0000	⊕	0000	=	0000
1010	⊕	0010	=	1000
0010	⊕	1110	=	1100
1110	⊕	1001	=	0111
1001	⊕	0100	=	1101
0100	⊕	0111	=	0011
0111	⊕	0001	=	0110
0001	⊕	1000	=	1001
1000	⊕	0011	=	1011
0011	⊕	0110	=	0101
0110	⊕	1100	=	1010
1100	⊕	1101	=	0001
1101	⊕	1111	=	0010
1111	⊕	1011	=	0100
1011	⊕	0101	=	1110
0101	⊕	1010	=	1111

The results now are:

$$|P_2 \cap H_3| = 0 \quad |P_i \cap H_j| = 4 \text{ for the other 15 sets}$$

$$NP_2 = 12 \text{ and } NP_1 = NP_3 = NP_4 = 16$$

Thus: $$\frac{\sum NP_i}{128} = \frac{60}{128} = 0.47$$

and the avalanching, while not strict, is closer to it by some visceral measure than in the second example.

METHODS OF NON-LINEAR DYNAMIC SUBSTITUTION

1. Using the Generating Function and Base Set, generate the set of Linear Orthomorphism Equations:

Equation #					
1	x_m	$\oplus$	x_1	$=$	x_{1-p}
2	x_1	$\oplus$	x_2	$=$	x_{2-p}
.	.	.	.	.	.
.	.	.	.	.	.
j	x_{j-1}	$\oplus$	x_j	$=$	x_{j-p}
.	.	.	.	.	.
.	.	.	.	.	.
m	x_{m-1}	$\oplus$	x_m	$=$	x_{m-p}

Where $m = 2^n - 1$

2. Determine q such that $x_q = x_1 \oplus x_2 \oplus x_3$

3. Based on q and the factors of m determine:

a) Which of the following two sets of Equations to be nonlinearized for each iteration of

a = 0, 1, 2, .. I:

Set 1 = B+ad, B+1+ad, B+2+ad and B+q-1+ad

Set 2 = B+ad, B+1+ad, B+2+ad, B+3+ad, B+q-1+ad, B+q+ad

Where B is the Beginning Equation and may be any number between 1 and m, inclusive, and d is defined below.

b) The spacing, d, between successive sets of nonlinearized equations (e.g., equations B+3d, B+1+3d, B+2+3d and B+q-1+3d will be nonlinearized in one iteration and then, in the next iteration, equations B+4d, B+1+4d, B+2+4d and B+q-1+4d will be nonlinearized).

Note: In some instances, both Set 1 and Set 2 will be used alternately or in some other pattern.

4. Starting with a = 0 and stopping when either a = I (the input number of iterations) or when the process attempts to work on rows already nonlinearized, iteratively perform the nonlinearizing algorithm defined below:

If Set 1 is being nonlinearized, add vectorially to each of the Set 1 equations,

$$(x_{B+ad} \oplus x_{B+1+ad}) \oplus (x_{B+ad} \oplus x_{B+1+ad}) = \theta.$$

If Set 2 is being nonlinearized,

add vectorially to equations B+ad and B+g-1+ad,

$$(x_{B+ad} \oplus x_{B+1+ad}) \oplus (x_{B+ad} \oplus x_{B+1+ad}) = \theta;$$

add vectorially to equations B+1+ad and B+2+ad,

$(x_{B+ad} \oplus x_{B+2+ad}) \oplus (x_{B+ad} \oplus x_{B+2+ad}) = \theta$; and

add vectorially to equations B+3+ad and B+Q+ad,

$$(x_{B+1+ad} \oplus x_{B+2+ad}) \oplus (x_{B+1+ad} \oplus x_{B+2+ad}) = \theta.$$

Note: In all cases, if any equation number or index to x is greater than m, the value of m minus that number will be used instead of that number (e.g., if B+1+ad is greater than m, m - (B+1+ad) will be used instead).

5. Generate the naturally ordered Encryption Look-up Table and De-encryption Look-up Table which reflect the Orthomorphic Encryption Permutation and the Orthomorphic De-encryption Permutation resulting from the Nonlinear Orthomorphism Equations generated in step 4. The Nonlinear Dynamic Substitution Device is comprised of these look-up tables plus mechanisms for changing the transformation fixed point.

As an alternative to the Look-up Table, two sets of modified equations can be used:

For encryption, the clear text is column 1, which is encrypted into the cipher text in column 3:

$$\begin{array}{ccccc}
1 & & 2 & & 3 \\
y_m & \oplus & y_1 & = & z_1 \\
y_1 & \oplus & y_2 & = & z_2 \\
. & . & . & . & . \\
. & . & . & . & . \\
y_{j-1} & \oplus & y_j & = & z_j \\
. & . & . & . & . \\
. & . & . & . & . \\
y_{m-1} & \oplus & y_m & = & z_m
\end{array}$$

Columns 1 and 2 are in the same order but shifted by one position. The numbers in column 3 are in a different order.

For decryption, the cipher text in column 3 is de-encrypted into the clear text in column 1

$$\begin{array}{ccccc}
1 & & 2 & & 3 \\
V_1 & = & w_1 & \oplus & w_m \\
V_2 & = & w_2 & \oplus & w_1 \\
. & . & . & . & . \\
. & . & . & . & . \\
V_j & = & w_j & \oplus & w_{j-1} \\
. & . & . & . & . \\
. & . & . & . & . \\
V_m & = & w_m & \oplus & w_{m-1}
\end{array}$$

The order of the rows of equations have now been arranged leaving the individual equations unchanged, so that columns 2 and 3 are in the same order, but shifted by one position, the numbers in column 1 are now in a different order from those in columns 2 and 3.

CLAIMS

I claim:

1. A method of encryption by substituting for any one of the 2^n unique clear text blocks of n bit binary numbers an associated unique encrypted block of n bit binary numbers comprising the steps of;

(a) finding a first matrix of 2^n equations, each equation representing the modulo 2 addition of one of the 2^n clear text blocks with a unique one of 2^n n bit numbers to provide an associated unique intermediate n bit block, all of the equations in the first matrix of 2^n equations being characterized by the vector sum modulo 2 of any number of the equations also being one of the equations in the first matrix, the equations including the null equation $\Theta \oplus \Theta = \Theta$ and the remaining $2^n - 1$ equations being orderable as follows;

Equation #					
1	x_m	$\oplus$	x_1	=	x_{1-p}
2	x_1	$\oplus$	x_2	=	x_{2-p}
.	.	.	.	.	.
.	.	.	.	.	.
j	x_{j-1}	$\oplus$	x_j	=	x_{j-p}
.	.	.	.	.	.
.	.	.	.	.	.
m	x_{m-1}	$\oplus$	x_m	=	x_{m-p}

where $m = 2^n - 1$

(b) modifying a plurality of the nonzero $2^n - 1$ equations in the first matrix of 2^n equations to provide a second matrix of 2^n equations, the plurality of equations being modified so that the modified plurality of equations collectively map the same clear text blocks to the same unique n bit intermediate blocks as the corresponding unmodified equations, but each in a different manner so that each of the modified equations is not the sum modulo 2 of any number of the equations in the unmodified first set; and,

(c) for each clear text block to be encrypted, adding modulo 2 to that block, the unique one of the 2^n n bit numbers associated therewith in accordance with the associated equation of the second matrix of 2^n equations to obtain the encrypted block.

2. The method of claim 1 wherein step (b) further comprises the step of adding modulo 2 an offset to each of the blocks in the first and second columns of the first matrix after modifying a plurality of the nonzero $2^n - 1$ equations in the first matrix of 2^n equations to provide the second matrix.

3. The method of claims 1 or 2 wherein in step (b), a plurality of the equations in the second set which are modified comprise three consecutive equations of the first set when ordered as set fourth in step (b), and a fourth equation which is the vector sum modulo 2 of the three

consecutive equations, the equations being modified by adding vectorially to each of the four equations, the following equation:

$$(x_1 \oplus x_2) \oplus (x_1 \oplus x_2) = \Theta,$$

wherein for purposes of the foregoing equations the first matrix of equations are ordered so that the first of the three consecutive equations is

$$x_m \oplus x_1 = x_{1-p}.$$

4. The method of claims 1 or 2 wherein in step (b), a plurality of the equations in the second set which are modified comprise four consecutive equations of the first set when ordered as set fourth in step (b), a fifth equation which is the vector sum modulo 2 of the first three of the four consecutive equations, and a sixth equation which is the vector sum modulo 2 of the last three of the four consecutive equations, the equations being modified by adding vectorially to the six equations, the following equations;

to equations 1 and q $(x_1 \oplus x_2) \oplus (x_1 \oplus x_2) = \Theta$

to equations 2 and 3 $(x_1 \oplus x_3) \oplus (x_1 \oplus x_3) = \Theta$

to equations 4 and q + 1 $(x_2 \oplus x_3) \oplus (x_2 \oplus x_3) = \Theta$

wherein for purposes of the foregoing equations the first matrix of equations are ordered so that the first of the four consecutive equations is

$$x_m \oplus x_1 = x_{1-p}.$$

$$\begin{array}{lllll}
000 & \oplus & 000 & = & 000 \\
001 & \oplus & 011 & = & 010 \\
010 & \oplus & 110 & = & 100 \\
011 & \oplus & 100 & = & 111 \\
100 & \oplus & 111 & = & 011 \\
101 & \oplus & 001 & = & 100 \\
110 & \oplus & 101 & = & 011 \\
111 & \oplus & 010 & = & 101
\end{array}$$

FIG. 1

$$\begin{array}{lllll}
000 & \oplus & 000 & = & 000 \\
001 & \oplus & 111 & = & 110 \\
010 & \oplus & 011 & = & 001 \\
011 & \oplus & 100 & = & 111 \\
100 & \oplus & 110 & = & 010 \\
101 & \oplus & 001 & = & 100 \\
110 & \oplus & 101 & = & 011 \\
111 & \oplus & 010 & = & 101
\end{array}$$

FIG. 2

$$\begin{array}{lllll}
000 & \oplus & 000 & = & 000 \\
011 & \oplus & 100 & = & 111 \\
100 & \oplus & 110 & = & 010 \\
110 & \oplus & 101 & = & 011 \\
101 & \oplus & 001 & = & 100 \\
001 & \oplus & 111 & = & 110 \\
111 & \oplus & 010 & = & 101 \\
010 & \oplus & 011 & = & 001
\end{array}$$

FIG. 3

$$\begin{array}{lllll}
000 & \oplus & 000 & = & 000 \\
110 & \oplus & 100 & = & 010 \\
101 & \oplus & 110 & = & 011 \\
001 & \oplus & 101 & = & 100 \\
111 & \oplus & 001 & = & 110 \\
010 & \oplus & 111 & = & 101 \\
011 & \oplus & 010 & = & 001 \\
100 & \oplus & 011 & = & 111
\end{array}$$

FIG. 4

$$\begin{array}{lllll}
000 & = & 000 & \oplus & 000 \\
110 & = & 100 & \oplus & 010 \\
101 & = & 110 & \oplus & 011 \\
001 & = & 101 & \oplus & 100 \\
111 & = & 001 & \oplus & 110 \\
010 & = & 111 & \oplus & 101 \\
011 & = & 010 & \oplus & 001 \\
100 & = & 011 & \oplus & 111
\end{array}$$

FIG. 5

$$\begin{array}{lllll}
001 & \oplus & 001 & = & 000 \\
111 & \oplus & 101 & = & 010 \\
100 & \oplus & 111 & = & 011 \\
000 & \oplus & 100 & = & 100 \\
110 & \oplus & 000 & = & 110 \\
011 & \oplus & 110 & = & 101 \\
010 & \oplus & 011 & = & 001 \\
101 & \oplus & 010 & = & 111
\end{array}$$

FIG. 10

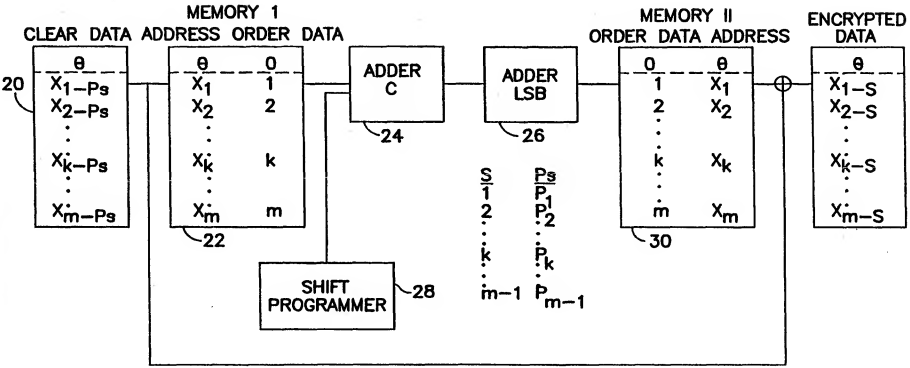

MEMORY 1
CLEAR DATA
ADDRESS
ORDER
DATA
MEMORY II
ORDER
DATA
ADDRESS
ENCRYPTED DATA
θ
X_1-Ps
X_2-Ps
X_k-Ps
X_m-Ps
20
θ
X_1
X_2
X_k
X_m
0
1
2
k
m
22
ADDER C
24
ADDER LSB
26
SHIFT PROGRAMMER
28
S
1
2
k
m−1
Ps
P_1
P_2
P_k
P_{m-1}
0
1
2
k
m
θ
X_1
X_2
X_k
X_m
30
θ
X_1-S
X_2-S
X_k-S
X_m-S

FIG. 6

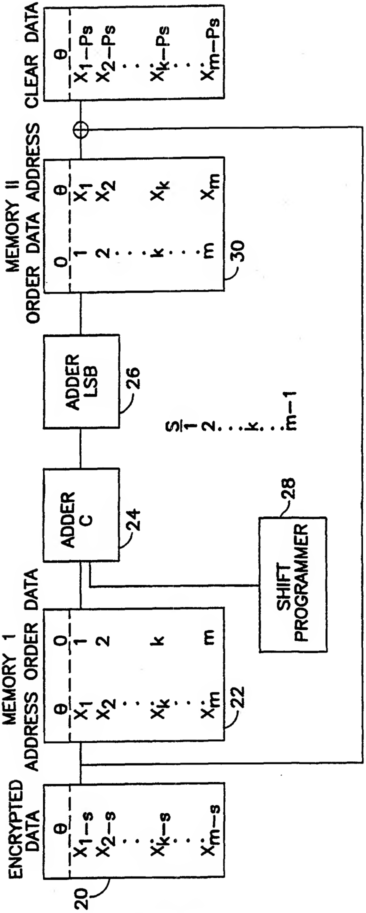
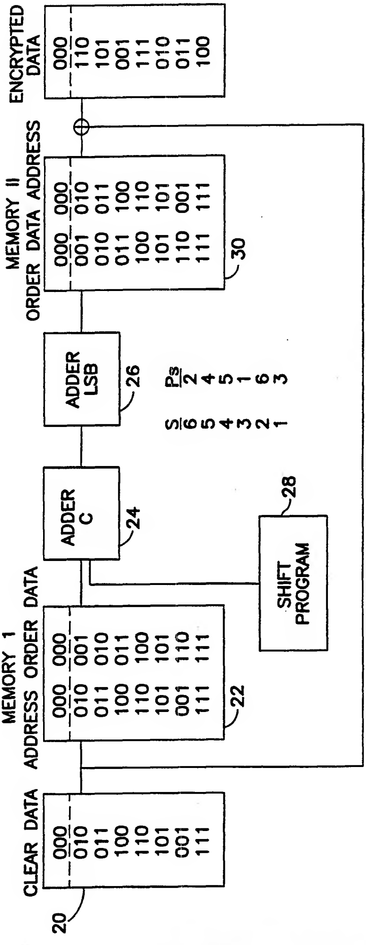

ENCRYPTED DATA
θ
X1−s
X2−s
Xk−s
Xm−s
20
MEMORY 1
ADDRESS ORDER DATA
θ
X1
X2
Xk
Xm
0
1
2
k
m
22
ADDER C
24
SHIFT PROGRAMMER
28
ADDER LSB
26
S
1
2
k
m−1
MEMORY II
ORDER DATA ADDRESS
0
1
2
k
m
θ
X1
X2
Xk
Xm
30
CLEAR DATA
θ
X1−Ps
X2−Ps
Xk−Ps
Xm−Ps

FIG. 7



CLEAR DATA
000
010
011
100
110
101
001
111

ADDRESS	ORDER DATA
000	000
010	001
011	010
100	011
110	100
101	101
001	110
111	111

S	Ps
6	2
5	4
4	5
3	1
2	6
1	3

ORDER DATA	ADDRESS
000	000
001	010
010	011
011	100
100	110
101	101
110	001
111	111

ENCRYPTED DATA
000
110
101
001
111
010
011
100

FIG. 8

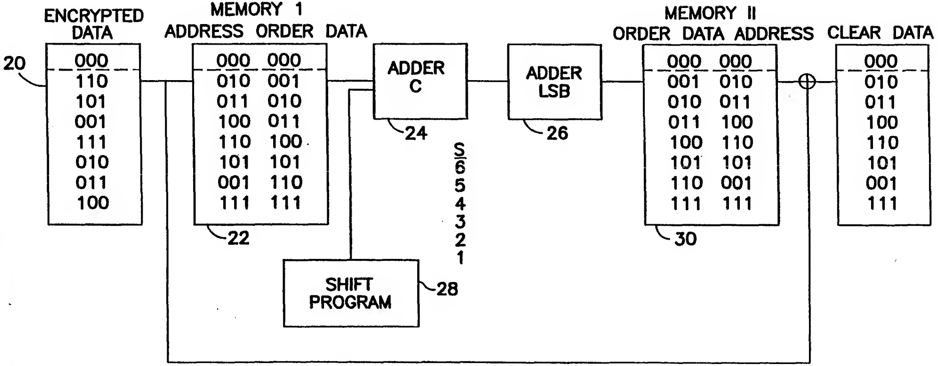

ENCRYPTED DATA
20
000
110
101
001
111
010
011
100
MEMORY 1
ADDRESS ORDER DATA
000 000
010 001
011 010
100 011
110 100
101 101
001 110
111 111
22
ADDER C
24
SHIFT PROGRAM
28
S
6
5
4
3
2
1
ADDER LSB
26
MEMORY II
ORDER DATA ADDRESS
000 000
001 010
010 011
011 100
100 110
101 101
110 001
111 111
30
CLEAR DATA
000
010
011
100
110
101
001
111

FIG. 9

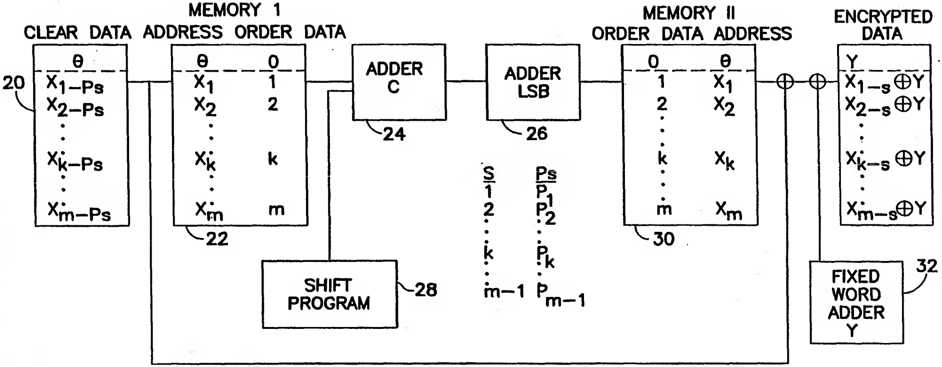

CLEAR DATA
θ
X1–Ps
X2–Ps
Xk–Ps
Xm–Ps
20
MEMORY 1
ADDRESS ORDER DATA
θ 0
X1 1
X2 2
Xk k
Xm m
22
ADDER C
24
ADDER LSB
26
SHIFT PROGRAM
28
S Ps
1 P1
2 P2
k Pk
m–1 Pm–1
MEMORY II
ORDER DATA ADDRESS
0 θ
1 X1
2 X2
k Xk
m Xm
30
ENCRYPTED DATA
Y
X1–s ⊕Y
X2–s ⊕Y
Xk–s ⊕Y
Xm–s ⊕Y
FIXED WORD ADDER Y
32

FIG. 11

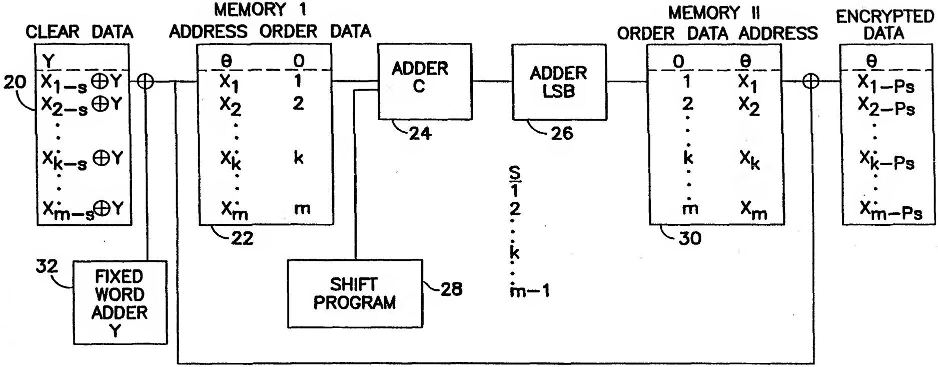
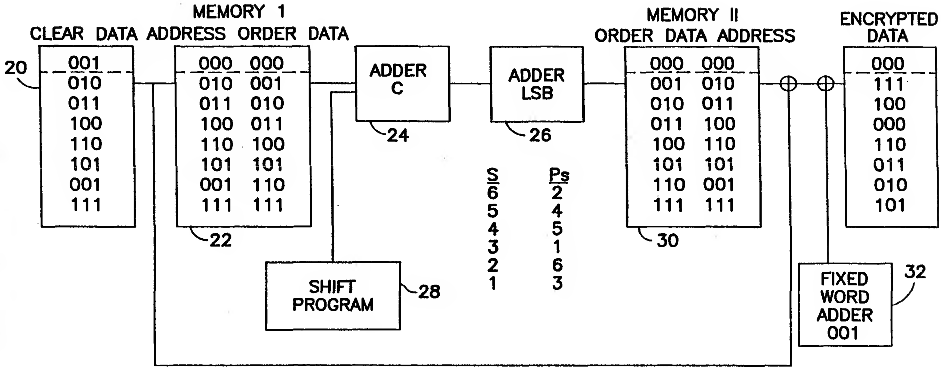

CLEAR DATA
Y
X1-s ⊕Y
X2-s ⊕Y
Xk-s ⊕Y
Xm-s ⊕Y
20
MEMORY 1
ADDRESS ORDER DATA
θ 0
X1 1
X2 2
Xk k
Xm m
22
ADDER C
24
ADDER LSB
26
MEMORY II
ORDER DATA ADDRESS
0 θ
1 X1
2 X2
k Xk
m Xm
30
ENCRYPTED DATA
θ
X1-Ps
X2-Ps
Xk-Ps
Xm-Ps
FIXED WORD ADDER Y
32
SHIFT PROGRAM
28
S
1
2
k
m-1

FIG. 12



CLEAR DATA
001
010
011
100
110
101
001
111

MEMORY 1

ADDRESS	ORDER DATA
000	000
010	001
011	010
100	011
110	100
101	101
001	110
111	111

S	Ps
6	2
5	4
4	5
3	1
2	6
1	3

MEMORY II

ORDER DATA	ADDRESS
000	000
001	010
010	011
011	100
100	110
101	101
110	001
111	111

ENCRYPTED DATA
000
111
100
000
110
011
010
101

FIG. 13

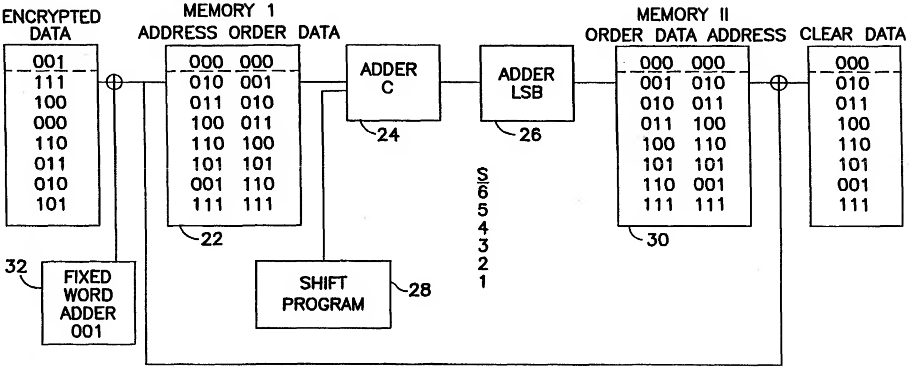

ENCRYPTED DATA
001
111
100
000
110
011
010
101
MEMORY 1
ADDRESS ORDER DATA
000 000
010 001
011 010
100 011
110 100
101 101
001 110
111 111
22
ADDER C
24
ADDER LSB
26
SHIFT PROGRAM
28
S
6
5
4
3
2
1
MEMORY II
ORDER DATA ADDRESS
000 000
001 010
010 011
011 100
100 110
101 101
110 001
111 111
30
CLEAR DATA
000
010
011
100
110
101
001
111
32
FIXED WORD ADDER 001

FIG. 14

$000 \oplus 000 = 000$

$101 \oplus 001 = 100$

$001 \oplus 111 = 110$

$111 \oplus 010 = 101$

$010 \oplus 011 = 001$

$011 \oplus 100 = 111$

$100 \oplus 110 = 010$

$110 \oplus 101 = 011$

FIG. 15

OFFSET = 0101

$0101 \oplus 0101 = 0000$

$1111 \oplus 0111 = 1000$

$0111 \oplus 0100 = 0011$

$0100 \oplus 0010 = 0110$

$0010 \oplus 1011 = 1001$

$1011 \oplus 1010 = 0001$

$1010 \oplus 1000 = 0010$

$1000 \oplus 0011 = 1011$

$0011 \oplus 0110 = 0101$

$0110 \oplus 1100 = 1010$

$1100 \oplus 0001 = 1101$

$0001 \oplus 1101 = 1100$

$1101 \oplus 1001 = 0100$

$1001 \oplus 1110 = 0111$

$1110 \oplus 0000 = 1110$

$0000 \oplus 1111 = 1111$

FIG. 17

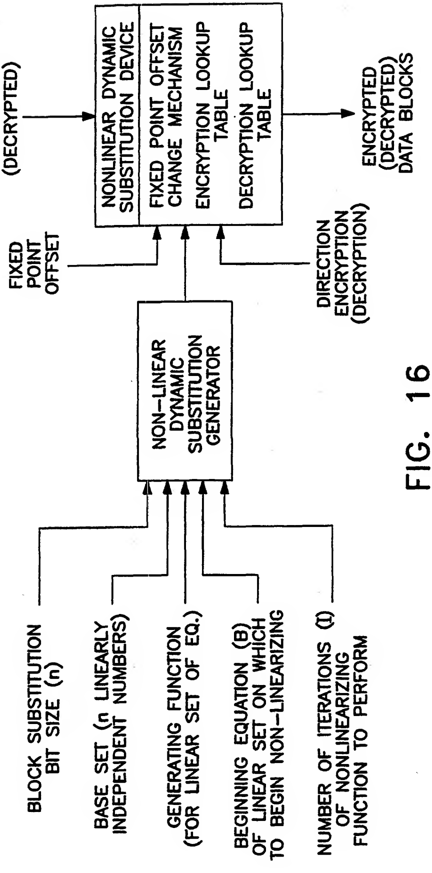

DATA BLOCKS TO BE ENCRYPTED (DECRYPTED)
NONLINEAR DYNAMIC SUBSTITUTION DEVICE
FIXED POINT OFFSET CHANGE MECHANISM
ENCRYPTION LOOKUP TABLE
DECRYPTION LOOKUP TABLE
ENCRYPTED (DECRYPTED) DATA BLOCKS
FIXED POINT OFFSET
DIRECTION ENCRYPTION (DECRYPTION)
NON-LINEAR DYNAMIC SUBSTITUTION GENERATOR
BLOCK SUBSTITUTION BIT SIZE (n)
BASE SET (n LINEARLY INDEPENDENT NUMBERS)
GENERATING FUNCTION (FOR LINEAR SET OF EQ.)
BEGINNING EQUATION (B) OF LINEAR SET ON WHICH TO BEGIN NON-LINEARIZING
NUMBER OF ITERATIONS (I) OF NONLINEARIZING FUNCTION TO PERFORM

FIG. 16

PCT/US92/06497

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :H04K 1/04
US CL :380/37
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 380/28,36,37,42,49

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

NONE

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US,A, 4,195,200 (FEISTEL), 25 MARCH 1980.	
A	US,A, 4,322,577 (BRANDSTROM), 30 MARCH 1982.	
A	US,A, 4,520,232 (WILSON), 28 MAY 1985.	
A	US,A, 4,685,132 (BISHIP ET AL.), 04 AUGUST 1987.	
AP	US,A, 5,038,376 (MITTENTHAL), 06 AUGUST 1991.	

□ Further documents are listed in the continuation of Box C. □ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 02 SEPTEMBER 1992	Date of mailing of the international search report 0NOV 199
Name and mailing address of the ISA/ US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer TOD R. SWANN
Facsimile No. NOT APPLICABLE	Telephone No. (703) 308-0475